TUGAS AKHIR – KI141502

DETEKSI KELELAHAN MENTAL BERBASIS GELOMBANG OTAK MENGGUNAKAN TRANSFORMASI FOURIER DAN SUPPORT VECTOR MACHINE

FARRAS KINAN
NRP 5111100161

Dosen Pembimbing I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

UNDERGRADUATE THESES – KI141502

MENTAL FATIGUE DETECTION BASED ON EEG USING FOURIER TRANSFORM AND SUPPORT VECTOR MACHINE

FARRAS KINAN
NRP 5111100161

Supervisor I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

Supervisor II
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2015

LEMBAR PENGESAHAN

DETEKSI KELELAHAN MENTAL BERBASIS GELOMBANG OTAK MENGGUNAKAN TRANSFORMASI FOURIER DAN SUPPORT VECTOR MACHINE

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
FARRAS KINAN
NRP : 5111 100 161

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D. ..... (Pembimbing 1)
   NIP: 194908231976032001
2. Dini Adni Navastara, S.Kom., M.Sc. ..... (Pembimbing 2)
   NIP: 5100201405003

SURABAYA
JUNI, 2015

DETEKSI KELELAHAN MENTAL BERBASIS GELOMBANG OTAK MENGGUNAKAN TRANSFORMASI FOURIER DAN SUPPORT VECTOR MACHINE

Nama Mahasiswa	**:**	**FARRAS KINAN**
NRP	**:**	**51111000161**
Jurusan	**:**	**Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**:**	**Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.**
Dosen Pembimbing 2	**:**	**Dini Adni Navastara, S.Kom., M.Sc.**

Abstrak

Dewasa ini, penyebab kematian akibat kecelakaan lalu lintas menjadi sangat tinggi. Salah satu faktor utama penyebab kecelakaan ini adalah kelelahan pengemudi. Hal ini dapat terjadi karena pengemudi kurang menyadari keadaan mentalnya yang sudah lelah. Tentu kelelahan mental dapat menyebabkan kurangnya konsentrasi saat mengemudi.

Kelelahan mental ini dapat dideteksi dengan menganalisis gelombang otak melalui sinyal EEG dari pengemudi. Analisis gelombang otak ini dapat dilakukan dengan berbagai metode. Pada Tugas Akhir kali ini, penulis melakukan deteksi kelelahan mental berbasis gelombang otak menggunakan transformasi *fourier* dan *support vector machine*. Data sinyal EEG akan dilakukan ekstraksi fitur menggunakan Transformasi *Fourier*. Lalu, hasil ekstraksi ini akan digunakan untuk proses klasifikasi dengan metode *Support Vector Machine.*

Berdasarkan hasil percobaan, klasifikasi kelelahan mental menggunakan *Support Vector Machine* dengan *linear*

kernel didapat rata-rata akurasi sebesar 85%. Dari hasil yang diperoleh dari uji coba dapat disimpulkan bahwa kombinasi metode yang digunakan pada Tugas Akhir ini mampu melakukan klasifikasi sinyal EEG untuk kelelahan mental.

Kata kunci: Kelelahan Mental, EEG, Transformasi Fourier, Support Vector Machine.

MENTAL FATIGUE DETECTION BASED ON EEG USING FOURIER TRANSFORM AND SUPPORT VECTOR MACHINE

Student's Name	**:**	**FARRAS KINAN**
Student's ID	**:**	**5111100161**
Department	**:**	**Teknik Informatika FTIF-ITS**
First Advisor	**:**	**Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.**
Second Advisor	**:**	**Dini Adni Navastara, S.Kom., M.Sc.**

Abstract

These days, deaths caused by road accidents raise at higher rates. One of the main factors that influences these situations are driver drowsiness. An accident can happen if the driver does not realize that he or she is tired. Mental fatigue can the drivers lose their concentration.

Mental fatigue can be detected by analyzing the brain wave via EEG signal from the driver. There are many methods that could be applied to analyze brain wave. In this undergraduate theses, the writer propose mental fatigue detection based on EEG using Fourier Transform and Support Vector Machine. EEG signal's features are extracted using Fourier Transform. Then, the result of feature extraction process will be used for classification process using Support Vector Machine.

By applying these methods, mental fatigue classification using Support Vector Machine with linear kernel, the writer obtains result the average of accuracy of 85%. From the results, it can be the concluded that

combination of methods that used in this undergraduate theses can do the classification of the mental fatigue EEG signal.

Keywords: Mental Fatigue, EEG, Fourier Transform, Support Vector Machine.

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"DETEKSI KELELAHAN MENTAL BERBASIS GELOMBANG OTAK MENGGUNAKAN TRANSFORMASI FOURIER DAN SUPPORT VECTOR MACHINE".**

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Ayah dan Mama yang telah memberikan dukungan moral dan material serta do'a yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
3. Ibu Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D. selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
4. Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.

5. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
6. Teman-teman 💪🎓: Adhi, Dafi, Dina, Monika, Rizaldi, dan Rizka yang telah memberi semangat, canda tawa, dan hiburan serta liburan.
7. Teman-teman Lab KCV: Hayam, Didit, Petrus, dan Aisha yang telah membantu memberi solusi masalah yang dihadapi penulis pada pengerjaan Tugas Akhir ini.
8. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2015

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dewasa ini, penyebab kematian akibat kecelakaan lalu lintas menjadi sangat tinggi. Salah satu faktor utama penyebab kecelakaan ini adalah kelelahan pengemudi [**1**]. Hal ini dapat terjadi karena pengemudi kurang menyadari keadaan mentalnya yang sudah lelah. Tentu kelelahan mental dapat menyebabkan kurangnya konsentrasi saat mengemudi.

Kelelahan mental ini dapat dideteksi dengan menganalisis gelombang otak melalui sinyal EEG dari pengemudi. Analisis gelombang otak ini dapat dilakukan dengan berbagai metode. Pada Tugas Akhir kali ini, penulis melakukan deteksi kelelahan mental berbasis gelombang otak menggunakan transformasi *fourier* dan *support vector machine*. Data sinyal EEG akan dilakukan ekstraksi fitur menggunakan Transformasi *Fourier*. Lalu, hasil ekstraksi ini akan digunakan untuk proses klasifikasi dengan metode *Support Vector Machine*.

Dalam penerapan metode ini diharapkan dapat menunjukkan keadaan dari pengemudi, apakah pengemudi tersebut dalam keadaan lelah ataupun tidak. Selain itu, akan didapat pula kesimpulan metode apa yang tepat untuk melakukan analisis gelombang otak untuk mengetahui keadaan mental seseorang.

1.2 Rumusan Masalah

Berikut beberapa hal yang menjadi rumusan masalah dalam Tugas Akhir ini:

1. Bagaimana mengimplementasikan tahap *preprocessing* gelombang EEG menggunakan Transformasi *Fourier*?
2. Bagaimana menerapkan klasifikasi kelelahan mental dengan menggunakan metode *Support Vector Machine*?
3. Bagaimana menghitung akurasi hasil klasifikasi kelelahan mental?

1.3 Batasan Masalah

Permasalahan yang akan dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Implementasi dilakukan dengan Matlab.
2. Rekaman gelombang EEG dilakukan dengan *NeuroSky MindWave*.
3. Penentuan kelas hasil transformasi gelombang EEG dilakukan dengan mengimplementasikan *Support Vector Machine* dan penggunaan kernel.

1.4 Tujuan

Tujuan Tugas Akhir ini adalah untuk membuat sistem yang mampu menunjukkan kondisi mental seseorang dalam keadaan lelah atau tidak secara otomatis berdasarkan sinyal EEG menggunakan metode ekstraksi fitur *Fast Fourier Transform* dan metode klasifikasi *Support Vector Machine*. Selain itu, untuk mengetahui kinerja metode yang diusulkan.

1.5 Manfaat

Tugas Akhir ini diharapkan mampu membangun sebuah sistem yang dapat membantu para *Automaker* (Manufaktur Otomotif) dalam membangun sistem keamanan kendaraan yang dapat mengetahui kondisi mental pengemudi. Dengan begitu

angka korban kecelakaan akibat kelelahan pengemudi dapat diminimalkan.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

9. Penyusunan proposal Tugas Akhir.
   Proposal Tugas Akhir ini berisikan mengenai apa saja yang dibutuhkan, serta rumusan masalah yang ada dalam deteksi kelelahan mental berbasis gelombang otak menggunakan transformasi *fourier* dan *support vector machine*.

10. Studi literatur
    Tugas Akhir ini menggunakan literatur *paper* beserta artikel dari jurnal internasional bereputasi yaitu *Sciencedirect*. *Paper* yang digunakan sebagai acuan utama dan dasar dalam pengerjaan Tugas Akhir ini adalah "EEG-Based Mental Fatigue Measurement Using Multi-Class Support Vector Machines with Confidence Estimate".

11. Analisis dan desain perangkat lunak
    Pada tahap ini akan dilakukan analisis dan design perancangan aplikasi sesuai dengan tujuan yang dijabarkan. Kemudian disesuaikan dengan metode yang tepat, hal ini dimaksudkan agar nantinya ketika diimplementasikan ke dalam aplikasi dapat berjalan sesuai yang diharapkan.

12. Implementasi perangkat lunak
    Implementasi merupakan tahap untuk membangun metode tersebut. Untuk mengimplementasikan metode tersebut menggunakan MATLAB dan memerlukan beberapa aplikasi yang lain seperti Microsoft Excel untuk perhitungan akurasi.

13. Pengujian dan evaluasi

    Pada tahap ini dilakukan uji coba dengan menggunakan bermacam masukan untuk menguji aplikasi apakah telah berjalan sesuai dengan rancangan dan desain implementasi yang dibuat, mengamati kinerja sistem, serta mengidentifikasi kendala yang mungkin muncul.

14. Penyusunan buku Tugas Akhir.

    Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Perancangan Perangkat Lunak

Bab ini berisi tentang desain sistem yang disajikan dalam bentuk *pseudocode*.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *code* yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Kelelahan Mental

Kelelahan mental adalah kondisi dimana seseorang secara temporal tidak mampu mempertahankan performa kognitif yang optimal. Pada permulaan kelelahan mental muncul secara bertahap lama aktivitas kognitif, dan bergantung pada kemampuan kognitif individu, dan beberapa faktor lainnya seperti kurang tidur dan kesehatan secara keseluruhan. [**2**]

Pada dasarnya, aktivitas mental yang dilakukan terus menerus akan mengakibatkan kekacauan. Banyak orang menunjukan tanda-tanda kelelahan mental pada siang hari di hari kerja. Mereka akan merasa pekerjaan akan semakin rumit, konsentrasi rendah, dan kecenderungan untuk melakukan kesalahan. Selain itu, menyelesaikan pekerjaan maupun belajar hingga larut malam dapat menyebabkan kelelahan mental, terhambatnya penyerapan informasi, sehingga menghasilkan kesalahan dan kecanggungan. [**3**]

2.2 Dataset Kelelahan Mental

Dataset kelelahan mental, didapat dengan melakukan rekaman pada seorang individu yang dilakukan berulang-ulang untuk menghindari hasil yang tidak konsisten jika dilakukan pada individu yang berbeda. [**4**].

Subjek yang direkam adalah seorang mahasiswa berusia 21 tahun yang memiliki aktivitas perkuliahan di siang hingga sore hari. Individu tersebut pergi ke kampus dari rumah dengan jarak tempuh sekitar 21 Km dengan menggunakan sepeda motor selama 45 menit dengan rute jalan protokol yang ramai terutama pada sore hari. Aktivitas subjek dilanjutkan dengan belajar pada malam hari hingga memasuki waktu tidur malam.

Rekaman gelombang otak untuk subjek tersebut dilakukan pada pagi hari dan malam hari agar menunjukkan perbedaan gelombang otak yang signifikan. Rekaman ini menggunakan alat bantu *NeuroSky MindWave*. Alat ini memiliki spesifikasi seperti pada Tabel 2.1.

Tabel 2.1 Spesifikasi MindWave

Jenis Spesifikasi	**Keterangan**
Sensor	Pasif
Jumlah Elektroda	2
Tipe Elektroda	Kering
Sampling Rate	512Hz
Spesifikasi lain	Nirkabel

Elektroda yang terdapat pada NeuroSky MindWave ini terdiri atas elektroda *receiver* dan elektroda *grounding*. Wujud NeuroSky MindWave ditunjukkan pada Gambar 2.1 Elektroda receiver ini akan bersentuhan dengan kulit kepala dimana fungsinya menerima gelombang sinyal dari otak. Karena sifat dari elektroda ini adalah elektroda kering, maka penggunaannya tidak perlu ditambah larutan *saline*. Elektroda *grounding* berbentuk seperti penjepit yang nantinya diletakkan pada cuping telinga yang fungsinya sebagai referensi *baseline voltage* dari tubuh manusia, elektroda ini sangat penting agar EEG tidak terganggu dengan aktivitas elektrik lain dalam tubuh.

Gambar 2.1 NeuroSky MindWave [5]

Hasil rekaman dengan menggunakan alat ini akan berupa gelombang otak dalam domain waktu dengan *sampling* sebesar 512Hz. Rekaman dilakukan selama beberapa menit lalu dilakukan pemotongan gelombang dengan durasi masing-masing selama 10 detik sebanyak 40 data dengan 20 data untuk setiap kelas. Pemotongan gelombang tersebut dilakukan dengan melakukan pengamatan secara visual dan pemilihan bagian gelombang yang akan digunakan.

2.3 *Electroencephalography*

Lapisan korteks otak terdiri dari *neuron-neuron* yang saling terhubung satu sama lain membentuk jaringan dan menerima input dari bagian otak yang lain. Aktivitas elektrik dalam bentuk rangsangan saraf yang dikirim maupun diterima oleh neuron korteks selalu terjadi meskipun pada saat tidur. Secara biologis, medis, dan hukum, ketidakadaan aktivitas tersebut menunjukkan kematian.

Aktivitas elektrik yang akan diukur menunjukkan aktivitas intrinsik dari *neuron* pada lapisan korteks otak dan informasi yang dikirimkan dari struktur subkortikal dan reseptor saraf. Keseluruhan aktivitas ini disebut dengan *Electroencephalogram* (EEG).

Sebuah elektroda EEG hanya akan merekam aktivitas dari area otak yang menempel dibawahnya. Meski begitu, elektroda tersebut menerima aktivitas dari ribuan *neuron*. Faktanya, 1 mm^2 korteks terdapat lebih dari 100.000 *neuron*. Jika input dari sebuah region tersinkronisasi dengan aktifitas elektrik terjadi pada saat yang bersamaan, hal tersebut menunjukan gelombang periodik sederhana dari EEG. Empat periode ritmik sederhana yang terekam dalam EEG adalah gelombang *alpha*, *beta*, *delta*, *theta*. [**6**]

Gelombang periode ritmik sederhana ini memiliki frekuensi yang berbeda. Frekuensi tersebut ditunjukkan pada Tabel 2.1. [**7**]

Tabel 2.2 Frekuensi EEG

Ritme	**Frekuensi**
delta	0,5 – 4 Hz
theta	4 – 8 Hz
alpha	8 – 13 Hz
beta	13 – 20 Hz

1. Delta
   Gelombang delta memiliki amplitude tertinggi diantara yang lain dan paling lambat. Ritme ini muncul secara dominan pada bayi usia satu tahun atau orang dewasa saat tidur dalam fase 3 atau 4. Gelombang delta juga muncul

pada aktivitas individu yang membutuhkan konsentrasi secara terus menerus.

2. Theta
   Gelombang theta termasuk gelombang dengan aktivitas lambat. Gelombang ini muncul pada anak-anak sebelum usia 13 tahun dan pada orang dewasa yang sedang tidur, kelelahan, maupun sedang bermeditasi.
3. Alpha
   Gelombang alpha pada umumnya muncul pada orang dewasa yang tersadar namun sedang berelaksasi dengan mata ditutup.
4. Beta
   Gelombang beta muncul pada individu yang sedang terjaga dan memperhatikan, mengingat sesuatu atau berpikir keras. Secara paradoks, gelombang ini juga muncul pada saat sesorang tertidur dalam fase *Rapid Eye Movement (REM)* atau tidur dengan mata bergerak cepat.

2.4 Transformasi Fourier

Teknik yang digunakan dalam analisis sinyal digital salah satunya untuk analisis EEG. ini termasuk analisis dengan wavelet dan fourier, yang ditujukan untuk mendapatkan ritme yang termasuk fase sinkroni (koherensi dan jeda fase) dan besarnya sinkroni (korelasi dan asimetri). [**8**]

Sinyal analog terdiri dari deret waktu tegangan mikro dari EEG, sampel digital dan tingkat sampling yang memadai untuk melakukan over-sampling sinyal. Amplifier EEG modern menggunakan sampel yang memadai untuk menguraikan EEG pada band medis tradisional dari DC ke 70 atau 100 Hz, dengan menggunakan tingkat sampel dari 250/256, 500/512, untuk lebih dari 1000 sampel per detik, tergantung pada aplikasi yang dimaksud.

Kuantisasi EEG pada Tugas Akhir ini akan dilakukan dengan transformasi *fourier*. Transformasi *Fourier* menguraikan deret waktu sinyal gelombang EEG menjadi tegangan dengan grafik frekuensi spektral yang biasa disebut dengan "Kekuatan Spektrum" dengan daya menjadi kuadrat dari besarnya EEG, dan besarnya menjadi rata-rata yang tidak terpisahkan dari amplitudo sinyal EEG, diukur dari puncak tertinggi (+) ke lembah terendah (-), dalam satu waktu. Panjang epoch menentukan resolusi frekuensi Fourier, dengan epoch 1 detik didapat resolusi 1 Hz, dan epoch 4 detik didapat ¼ Hz. [**8**]

Salah satu jenis transformasi *fourier* adalah *Fast Fourier Transform* (FFT). FFT diterapkan dalam beragam bidang dari pengolahan sinyal digital dan memecahkan persamaan diferensial parsial menjadi algoritma-algoritma untuk penggandaan bilangan integer dalam jumlah banyak. Ada pun kelas dasar dari algoritma FFT yaitu *decimation in time* (DIT) dan *decimation in frequency* (DIF). Garis besar dari kata Fast diartikan karena formulasi FFT jauh lebih cepat dibandingkan dengan metode perhitungan algoritma Transformasi Fourier sebelumnya. [**9**]

Metode FFT memerlukan sekitar 10000 operasi algoritma matematika untuk data dengan 1000 observasi, 100 kali lebih cepat dibandingan dengan metode sebelumnya. Penemuan FFT dan perkembangan personal komputer, teknik FFT dalam proses analisa data menjadi populer, dan merupakan salah satu metode baku dalam analisa data. Satu bentuk transformasi yang umum dimana $F(\omega)$ adalah sinyal dalam domain frekuensi dan domain waktu berupa $f(t)$, digunakan untuk merubah sinyal dari domain waktu ke domain frekuensi adalah Transformasi Fourier yang terdapat pada Persamaan 2.1.

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t} dt \qquad (2.1)$$

Sedangkan Persamaan 2.2 menunjukkan fungsi invers transform, yaitu mengembalikan domain frekuensi ke domain waktu.

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\, e^{i\omega t} d\omega \qquad (2.2)$$

FFT dalam pengolahan sinyal meliputi Periode dan frekuensi:

1 Periode

Secara umum periode didefinisikan sebagai waktu yang dibutuhkan untuk sebuah isyarat atau gelombang mencapai suatu gelombang penuh dan dapat menentukan nilai periodesitasnya. Perlu dicermati bahwa pengertian ini berlaku untuk isyarat monokromatis, isyarat yang dimaksud adalah gelombangnya bersifat tunggal, pasti memiliki sebuah periode. Dengan demikian isyarat itu dikenal dengan istilah periodis, pengamatan dapat dilakukan dengan memantau gelombang kita dapat mengetahui nilai nilai yang terkandung dalam isyarat serta periodenya.

2 Frekuensi

Ada periode, maka ada frekuensi diartikan sebagai jumlah gelombang yang terjadi dalam 1 detik. Frekuensi didefinisikan secara sederhana sebagai kebalikan dari waktu. Sehingga waktu yang satuannya adalah detik (second) akan menjadi Hertz (1-per second) hanya akan memiliki tepat satu nilai spektrum. Yang dikenal dengan spektrum frekuensi. Pengertian frekuensi ini juga berlaku untuk gelombang monokromatis. [**10**]

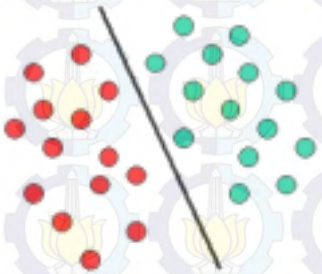
2.5 *Support Vector Machine*

Dalam *machine learning*, *Support Vector Machine* (SVM) termasuk model pembelajaran terarah *(supervised learning)* yang dikaitkan dengan analisis data dan pengenalan pola, digunakan juga untuk klasifikasi dan analisis regresi. [**11**]

Teknik SVM digunakan untuk menemukan *fungsi pemisah* (klasifier) yang optimal yang bisa memisahkan dua set data dari dua kelas yang berbeda. Penggunaan teknik machine learning tersebut, karena performansinya yang meyakinkan dalam memprediksi kelas suatu data baru. [**11**]

Salah satu contoh klasifier atau bidang pemisah yang biasa disebut dengan *hyperplane* ditunjukkan pada Gambar 2.2. Pada gambar tersebut ditunjukkan dua buah kelas yang dipisahkan dengan *linear hyperplane* sebagai bentuk yang paling sederhana pada SVM. Sebagai acuan, terdapat training set *D* yang dapat dinotasikan dengan Persamaan 2.3.

$$D = \{(x_i, y_i) | x_i \in R^p, y_i \in \{-1,1\}\}_{i=1}^{n} \quad (2.3)$$



Gambar 2.2 Dua buah kelas terpisah dengan *hyperplane* [12]

Nilai $y_i \in \{-1,1\}$ merupakan penunjuk *class* atau label yang bernilai 1 atau -1 dari kumpulan data x_i. *Hyperplane* yang terbaik merupakan *hyperplane* yang memisahkan data $y_i = 1$ dan

$y_i = -1$. *Hyperplane* yang memisahkan tersebut dapat dinotasikan dengan Persamaan 2.4.

$$w \cdot x - b = 0 \quad (2.4)$$

Titik data x_i yang termasuk kelas -1 dapat dinotasikan sebagai titik data yang memenuhi Pertidaksamaan 2.3 dan titik data x_i yang termasuk kelas 1 dapat dinotasikan dengan Pertidaksamaan 2.5.

$$w \cdot x - b \geq 1, y_i = 1 \quad (2.5)$$

$$w \cdot x - b \leq 1, y_i = -1 \quad (2.6)$$

Hyperplane yang ideal adalah *hyperplane* yang mempunyai margin yang maksimal dalam memisahkan kelas-kelas data. Sesuai dengan penerapan rumus perhitungan jarak antara titik dengan garis, maka jarak antara data terdekat dengan batas tengah *hyperplane* adalah $\frac{1}{\|w\|}$. Untuk memaksimalkan margin, maka nilai $\|w\|$ harus diminimalkan. Peminimalisasian nilai $\|w\|$ artinya harus menyelesaikan permasalahan *quadratic programming*. Permasalahan *quadratic programming* ini akan mencari titik minimal Persamaan 2.7 dengan memperhatikan batasan Persamaan 2.8.

$$f: \frac{1}{2}\|w\|^2 \quad (2.7)$$

$$y_i(x_i . w + b) - 1 \geq 0, \forall_1 \quad (2.8)$$

Permasalahan *quadratic programming* ini dapat dipecahkan dengan berbagai teknik komputasi. Salah satu metode pemecahannya adalah dengan menggunakan *Lagrange Multiplier*. *Lagrange Multiplier* menggunakan variabel α sehingga persamaan akan lebih mudah dihitung. Variabel α bernilai nol atau positif untuk persamaan 2.9

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_1 (y_i(x_i . w + b) - 1) \quad (2.9)$$

Nilai optimal dari Persamaan 2.9 dapat dihitung dengan meminimalkan L terhadap w dan b, dan memaksimalkan L terhadap α_1. Persamaan 2.9 juga dapat dimodifikasi sehingga hanya mengandung α_1 dengan memperhatikan nilai $L = 0$ pada titik optimal gradient menjadi Persamaan 2.10 dengan batasan Persamaan 2.11. Data yang berkorelasi dengan α_1 yang positif inilah yang disebut *support vector*.

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j \qquad (2.10)$$

$$\alpha_i \geq 0 \text{ dan } \sum_{i=1}^{l} \alpha_i y_i \qquad (2.11)$$

Terkadang, dalam beberapa kasus terdapat beberapa data pelatihan yang *error* atau terpisah dari yang lain baik karena sifatnya maupun dari nilai datanya. Kasus seperti ini biasa disebut dengan *non-separable case*. Kasus ini menyebabkan dua buah ruang masukan tidak dapat dipisahkan dengan sempurna sehingga pembentukan *decision boundary* dari kasus ini membutuhkan *soft margin*. *Soft margin* ini dapat menjadi toleransi data-data yang *error* tersebut sementara *margin* yang sebenarnya membentuk bidang/*plane* yang maksimum.

Pembentukan *decision boundary* akan membutuhkan *slack variable* (ξ) yang memberikan perkiraan error dari *decision boundary* terhadap data yang dilatih. Sementara untuk *data testing*, bidang yang terbentuk akan sangat lebar sehingga dapat diperkirakan *data testing* yang tidak berhasil diklasifikasi. Oleh karena itu fungsi objektif harus dimodifikasi dengan parameter *C* dan *k* agar dapat memberi penalty dari nilai fungsi dengan *slack variable*.

Dalam aplikasi *soft margin*, Persamaan 2.12 merupakan modifikasi Persamaan 2.8 dengan memasukkan *slack variable* $\xi_i(\xi_i > 0)$ dan Persamaan 2.13 adalah modifikasi Persamaan 2.7 dengan cara yang sama. Parameter *C* dan *k* biasanya ditentukan oleh pengguna dan kedua parameter ini merepresentasikan nilai penalty dari kesalahan klasifikasi terhadap data pelatihan.

$$y_i(x_i.w + b) \geq 1 - \xi, \forall_i \quad (2.12)$$

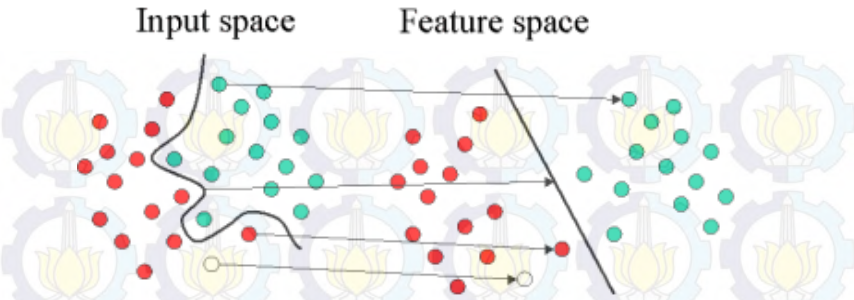
$$f: \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{N} \xi_i\right)^k \quad (2.13)$$

Pada dunia nyata, kasus *linearly separable* atau kasus data yang dapat dipisahkan secara linier seperti pada Gambar 2.2 jarang terjadi. Kasus yang terjadi pada umumnya bersifat nonlinier. Untuk menyelesaikan permasalahan nonlinier SVM dimodifikasi dengan memasukkan fungsi *kernel*.

Trik dalam mengerjakan nonlinier SVM adalah mentransformasi data dari ruang koordinat awal *x* menjadi ruang-ruang baru dengan fungsi $\emptyset(x)$ sehingga membentuk sebuah barasan linier yang dapat digunakan untuk memisahkan data-data yang diinginkan. Hal ini diterapkan agar selanjutnya dapat dilakukan metode pencarian batas bidang seperti pada proses linier SVM sebelumnya. Hal ini sejalan dengan teori Cover yang menyatakan *"Jika suatu transformasi bersifat non linear dan dimensi dari feature space cukup tinggi, maka data pada input space dapat dipetakan ke feature space yang baru, dimana pattern-pattern tersebut pada probabilitas tinggi dapat dipisahkan secara linear"*.

Ilustrasi dari konsep ini dapat dilihat pada Gambar 2.3. Pada gambar tersebut diperlihatkan input space pada data kelas hijau dan data kelas merah tidak dapat dipisahkan secara linier. Selanjutnya bagian *feature space* menunjukkan fungsi $\emptyset$ memetakan setiap data pada *input space* ke ruang *vector* baru yang berdimensi lebih tinggi, dimana kedua kelas dipisahkan secara *linear* oleh sebuah *hyperplane*. Persamaan 2.14 menunjukkan notasi matematika dari pemetaan ini.

$$\emptyset: \mathrm{R}^d \rightarrow R^q, d < q \quad (2.14)$$



Gambar 2.3 Ilustrasi *hyperplane* pada nonlinier SVM [12**]**

Pemetaan ini dilakukan dengan menjaga topologi data, dalam artian dua data yang berjarak dekat pada *input space* akan berjarak dekat juga pada *feature space*. Sebaliknya dua data yang berjarak jauh pada *input space* akan juga berjarak jauh pada *feature space*.

Selanjutnya proses pembelajaran pada SVM dalam menemukan titik-titik *support vector*, hanya bergantung pada *dot product* dari data yang sudah ditransformasikan pada ruang baru yang berdimensi lebih tinggi yaitu $\emptyset(x_i).\emptyset(x_j)$. Karena umumnya transformasi $\emptyset$ ini tidak diketahui, dan sangat sulit untuk dipahami secara mudah, maka perhitungan *dot product* tersebut sesuai teori Mercer dapat digantikan dengan fungsi *kernel* (x_i, x_j) yang mendefinisikan secara implisit transformasi $\emptyset$.

Fungsi pengganti transformasi tersebut kemudian lebih sering disebut dengan *kernel trick*. *Kernel trick* dapat disebut juga dengan metode untuk menghitung kesamaan dari ruang yang ditransformasi menggunakan atribut set awal dan nilai *dot product* dari dat. Secara umum, *kernel trick* dinotasikan dengan Persamaan 2.15

$$K(x_i, x_j) = \emptyset(x_i).\emptyset(x_j) \qquad (2.15)$$

Kernel trick memberikan berbagai kemudahan dalam proses pembelajaran SVM karena untuk menentukan *support vector*, kita hanya perlu mengetahui fungsi kernel yang dipakai dan tidak perlu mengetahui wujud dari fungsi non linear $\emptyset$. Beberapa jenis kernel yang sering dipakai ditunjukkan pada Tabel 2.3.

Tabel 2.3 Kernel yang umum dipakai dalam SVM

Jenis Kernel	Formula
Linear	$K(x_i, x_j) = x_i . x_j$
Radial Basis Function (RBF) / Gaussian	$K(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$
Polynomial	$K(x_i, x_j) = (x_i x_j + 1)^p$
Sigmoid	$K(x_i, x_j) = tanh(\alpha x_i . x_j + \beta)$

Selanjutnya hasil klasifikasi dari data x diperoleh dari Persamaan 2.16, 2.17, dan 2.18. Variabel *SV* pada Persamaan 2.16 dan Persamaan 2.17 merupakan subset dari *data training* yang terpilih sebagai *support vector* atau x_i yang berkorespondensi pada $\alpha_i \geq 0$. [**13**]

$$f(\emptyset(\mathrm{x})) = w . \emptyset(\mathrm{x}) + b \quad (2.16)$$

$$f(\emptyset(\mathrm{x})) = \sum_{i=1, x_i \in SV}^{l} \alpha_i y_i \emptyset(x) . \emptyset(x_i) + b \quad (2.17)$$

$$f(\emptyset(\mathrm{x})) = \sum_{i=1, x_i \in SV}^{l} \alpha_i y_i K(x . x_i) + b \quad (2.18)$$

2.6 *Cross Validation*

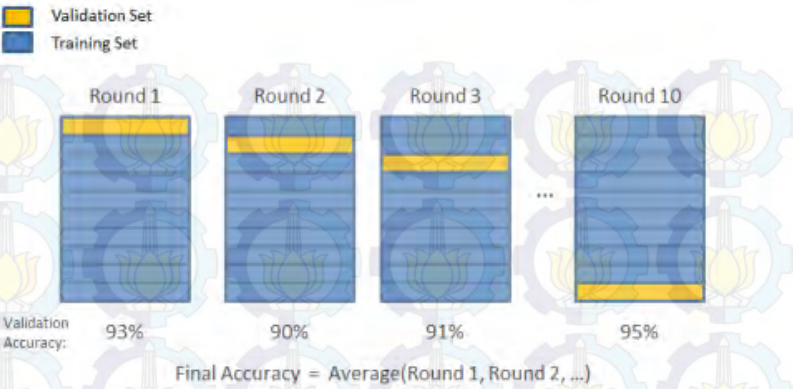
Cross Validation adalah suatu cara evaluasi performa prediksi dari suatu model statistik. Cara untuk melakukan validasi ini adalah dengan tidak menggunakan seluruh *dataset* pada saat melakukan pembelajaran. Beberapa data dihilangkan sebelum *training* dimulai. Ketika *training* selesai, data yang dihilangkan sebelumnya dapat digunakan untuk menguji performa dari model pembelajaran pada 'data baru'. Hal ini menjadi dasar dalam metode evaluasi keseluruhan kelas yang di sebut *cross validation*.

2.6.1 Holdout Method

Holdout method adalah jenis *cross validation* yang paling sederhana. *Dataset* dibagi menjadi dua set, yang disebut *training set* dan *testing set*. Fungsi aproksimasi mencocokan fungsi tersebut dengan *training set* saja. Lalu fungsi aproksimasi akan memprediksi nilai keluaran dari data dalam *testing set*. Hasil keluaran dari fungsi aproksimasi ini yang akan dihitung menjadi akurasi.

2.6.2 *K-Fold Cross Validation*

k-fold cross validation adalah salah satu cara untuk meningkatkan *holdout method*. Data set dibagi menjadi *k subset*, dan *holdout method* diulang sebanyak *k* kali. Setiak kali dijalankan, salah satu dati *k subset* digunakan menjadi *test set* dan *k – 1 subset* lainnya disatukan sebagai *training set*. Lalu, rata-rata *output* untuk semua *k* percobaan dihitung. Kelebihan metode ini adalah tidak terlalu berpengaruh bagaimana data dibagi. Setiap data hanya akan sekali diuji, dan manjadi *k – 1* kali sebagai *training set*. Ilustrasi *k-fold cross validation* akan ditunjukkan pada Gambar 2.4 dimana pada gambar tersebut memiliki nilai *k* sebesar 10. Akurasi yang ditunjukkan pada gambar tersebut hanya sebagai contoh. Pengujian ini dilakukan untuk mengetes semua data dan menghindari data *testing* yang terulang pada tahap *testing* dan pembagian data yang kurang bagus [**14**].



Gambar 2.4 ***10-fold cross validation***

BAB III
DESAIN PERANGKAT LUNAK

Pada bab tiga akan dijelaskan mengenain perancangan program yang dibuat untuk mencapai tujuan dari Tugas Akhir. Perancangan yang akan dijelaskan pada bab ini meliputi perancangan data dan perancangan proses. Selain itu akan dijelaskan juga desain metode secara umum pada sistem.

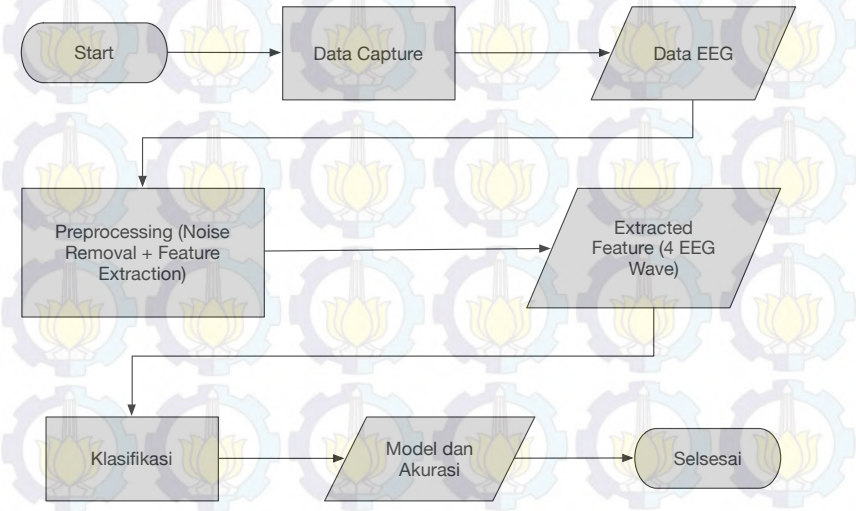
3.1 Desain Metode Secara Umum

Pada Tugas Akhir ini akan dibangun suatu sistem untuk melakukan deteksi kelelahan mental berbasis EEG. Proses-proses yang ada dalam sistem ini meliputi *data capture*, *preprocessing* dan klasifikasi. Tahap data capture merupakan tahap pertama dari sistem ini di mana perekaman gelombang otak dilakukan. Selanjutnya, tahap *preprocessing* dilakukan *noise* yang ada dalam sinyal hilang dan mengekstrak fitur dari gelombang tersebut agar data yang akan diklasifikasi dapat terklasifikasi dengan optimal. Tahap terakhir sistem ini adalah klasifikasi, yaitu menentukan data tersebut menunjukkan kelelahan mental atau tidak. Diagram alir sistem ditunjukkan oleh Gambar 3.1.

3.2 *Data Capture*

Dalam mengawali proses pengerjaan program, maka harus disiapkan terlebih dahulu data yang akan digunakan dalam proses pengerjaan. *Data capture* adalah tahapan yang penting untuk dilakukan. Pada tugas akhir kali ini, tahap *data capture* yang dilakukan adalah melakukan rekaman gelombang otak menggunakan NeuroSky MindWave. NeuroSky tidak memberikan SDK untuk mendapatkan gelombang EEG mentah yang diterima elektroda *receiver*. Namun, dengan mengakses port tertentu pada program ThinkGear Connector yang terdapat pada MindWave, gelombang EEG untuk Tugas Akhir ini dapat

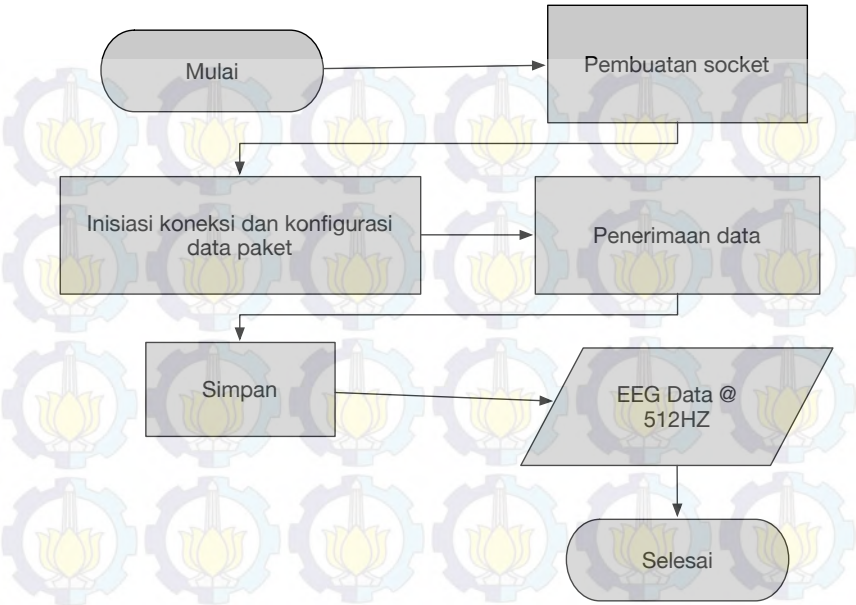
diperoleh. Gambar 3.2 menunjukkan tahapan untuk melakukan *data capture* secara lebih rinci.



Gambar 3.1 Diagram alir sistem

3.3 *Preprocessing*

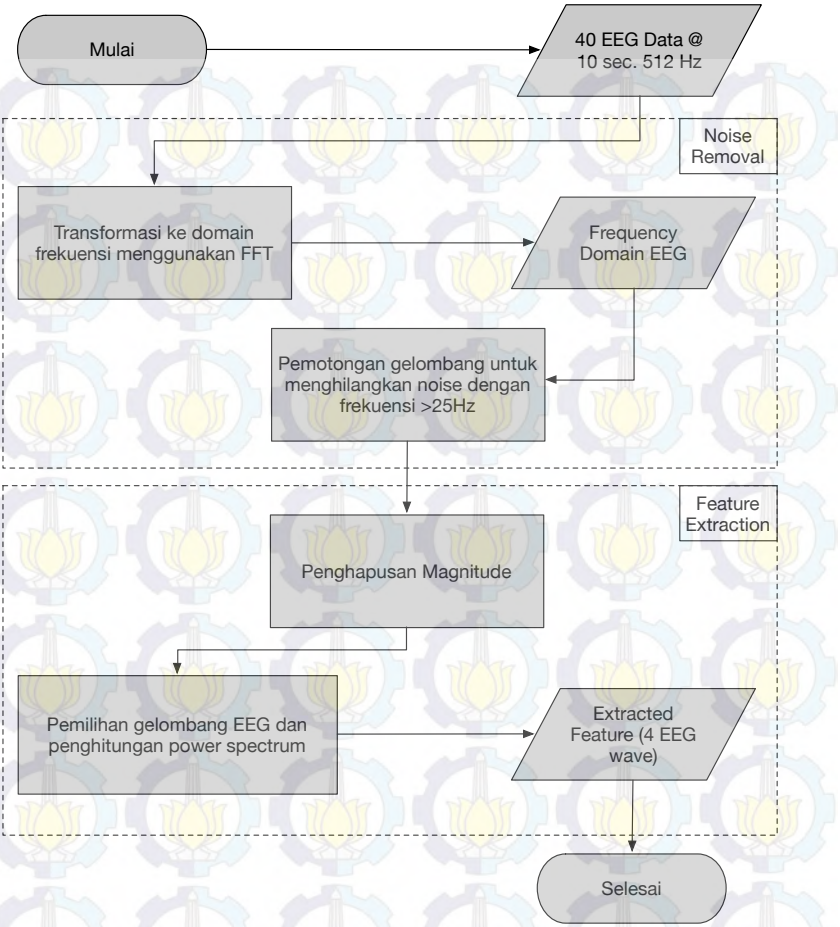
Preprocessing data ditujukan untuk mengubah data mentah menjadi bentuk yang lebih tepat untuk proses selanjutnya, yaitu klasifikasi. Pada tahap ini, *preprocessing* dilakukan dengan penghilangan *noise* pada gelombang EEG mentah dan kemudian dilakukan ekstraksi fitur, kedua langkah ini dilakukan dengan menerapkan algoritma *Fast Fourier Transform (FFT)*. Detil alur proses penghilangan noise dan ekstraksi fitur ini akan ditunjukkan pada Gambar 3.3.



Gambar 3.2 Diagram alir *Data Capture*

3.3.1 *Noise Removal*

Sinyal EEG dapat terkontaminasi *noise* atau artifak sinyal pada saat proses perekaman maupun proses transmisi data dari alat ke transmitter yang terhubung dengan komputer. Salah satu bentuk dari *noise* tersebut adalah pengaruh frekuensi kelistrikan. NeuroSky MindWave yang digunakan penulis berasal dari negara Amerika Serikat yang memiliki frekuensi kelistrikan sebesar 60Hz. Sedangkan negara Indonesia memiliki frekuensi yang berbeda dengan Amerika yaitu 50Hz, hal ini dapat menimbulkan noise atau artifak pada hasil rekaman sinyal.



Gambar 3.3 Diagram alir tahap preprocessing

Pada fungsi *noise removal* ini, sinyal akan difiltrasi untuk menghilangkan noise dengan mengaplikasikan fungsi FFT. Fungsi FFT akan mengubah sinyal menjadi domain frekuensi. Selanjutnya akan dipilih sinyal dengan frekuensi 0,5Hz-25Hz

untuk diolah dan sinyal selain itu dieliminasi. *Pseudocode* untuk implementasi fungsi *noise removal* terdapat pada Gambar 3.4.

Masukan	Sinyal EEG
Keluaran	Sinyal EEG dengan frekuensi <25Hz

```
1. sinyal ← sinyal EEG
2. frekuensi sinyal ← transform FFT (sinyal)
3. frekuensi sinyal >25 ← 0
4. sinyal baru ← invers transform FFT(frekuensi
   sinyal)
```

Gambar 3.4 *Pseudocode* fungsi *noise removal*

3.3.2 Feature Extraction

Proses selanjutnya setelah data sinyal dihilangkan *noise*-nya adalah mendapatkan fitur dari sinyal tersebut. Fitur-fitur tersebut dalam bentuk gelombang alpha, beta, delta, dan theta. Empat gelombang tersebut memiliki masing-masing frekuensi yang terdapat pada Tabel 2.2. Fitur yang akan digunakan untuk tahap selanjutnya akan berupa *power spectrum* dari masing-masing gelombang. **Gambar 3.5** adalah *pseudocode* dari fungsi *feature extraction*.

Masukan	Sinyal EEG dengan frekuensi <25Hz
Keluaran	Power spectrum alpha, beta, delta, theta

```
1. sinyal ← sinyal EEG
2. spektrum ← (absolute(sinyal))^2
3. alpha ← spektrum(frekuensi alpha)
4. beta ← spektrum(frekuensi beta)
5. delta ← spektrum(frekuensi delta)
6. theta ← spektrum(frekuensi theta)
7. feature ← [sum(alpha), sum(beta), sum(delta),
   sum(theta)]
```

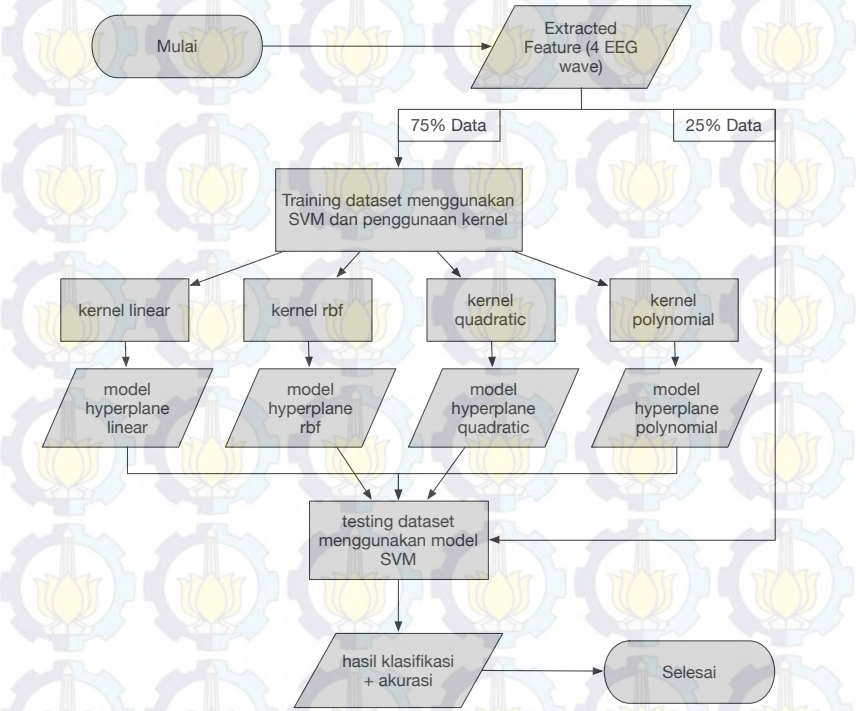
Gambar 3.5 *Pseudocode* fungsi *feature extraction*

Sebelum dapat diambil *power spectrum* dari masing-masing gelombang, gelombang hasil *noise removal* harus

dihilangkan *magnitude*-nya terlebih dahulu karena gelombang tersebut masih dalam bentuk *complex double*, sehingga haru dirubah ke dalam bentuk *double* agar dapat dioperasikan dan dihitung *power spectrum*-nya.

3.4 Klasifikasi

Data hasil dari tahapan preprocessing selanjutnya diklasifikasi untuk mendapatkan model training dan akurasi. Gambar 3.6 menunjukkan alur dari proses klasifikasi.



Gambar 3.6 Diagram alir proses klasifikasi

Pada awal tahapan ini data sinyal EEG yang sudah diekstrak fiturnya akan dilakukan pembagian untuk mendapatkan data *training* sebesar 75% dan sisanya 25% untuk data *testing*. Selanjutnya dilakukan proses *training* dengan 4 kernel SVM untuk mendapatkan *classifier model* yang nantinya digunakan untuk melakukan klasifikasi pada data *testing* agar didapat hasil klasifikasi berupa nilai akurasi.

3.4.1 Fungsi SVMTrain

Fungsi SVMTrain adalah fungsi yang digunakan untuk menghasilkan *model classifier* yang didapat dari proses training data. Model yang dibentuk oleh fungsi SVMTrain berupa sejumlah *vector* seperti yang mengandung informasi seperti *hyperplane*, *scaling*, *class*, *alpha*, dan *index*. Vektor *hyperplane* meunjukkan koordinat *hyperplane* berada. Vektor *scaling* merupakan acuan untuk normalisasi data. *Class* berisi kelas-kelas dari *data training* yang dimasukkan sebelumnya. Rincian fungsi SVMTrain yang sudah terdapat dalam Matlab akan ditunjukkan Gambar 3.7.

Masukan	Data training fitur sinyal EEG
Keluaran	*Model classifier*

```
1. data ← random(data fitur)
2. x ← data training
3. y ← kelas data training
4. model ← svmtrain (x , y , kernel)
```

Gambar 3.7 *Pseudocode* fungsi SVMTrain

3.4.2 Fungsi SVMClassify

Model classifier yang dihasilkan oleh fungsi SVMTrain selanjutnya digunakan untuk melakukan testing pada data fitur sinyal EEG. SVMClassify dari Matlab ini akan menghasilkan kelas dari data testing berdasarkan *hyperplane* yang terdapat pada *vector* dari *model classifier*. Gambar 3.8 menunjukkan *pseudocode* dari fungsi ini.

Masukan	Data testing dan Model classifier
Keluaran	Kelas data testing

```
1. xt ← data testing
2. hasil ← svmclassify ( model , xt )
```

Gambar 3.8 *Pseudocode* fungsi SVMClassify

3.4.3 Fungsi kCVal

Fungsi kCVal adalah fungsi yang digunakan untuk melakukan validasi dataset yang disediakan. Validasi dilakukan dengan menerapkan metode *k-fold cross validation*. Rincian fungsi kCVal akan ditunjukkan pada Gambar 3.9.

Masukan	Dataset fitur sinyal EEG
Keluaran	Akurasi & *Confusion matrix*

```
1. data ← random(data fitur)
2. groups ← kelas fitur
3. k ← jumlah partisi
4. cvfolds(k)
5. for i ← 1:k
6.     testindex ← cvfolds = 1
7.     trainindex ← !testindex
8.     model ← svmtrain (data(trainindex),
   groups(trainindex), kernel)
9.     res  svmclassify (model, data(textindex))
10. end
11. confmat ← classperf(countinmatrix)
12. accuracy ← classperf(correctrate)
```

Gambar 3.9 *Pseudocode* fungsi kCVal

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan digunakan untuk melakukan implementasi meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

1. Perangkat keras
   a. Prosesor: Intel® Core™ i5-5250U CPU @ 1.60GHz
   b. *Memory*(RAM): 4,00 GB
   c. Tipe Sistem: 64-bit Sistem operasi
2. Perangkat lunak
   a. Sistem operasi: *Windows 8.1 Pro* tervirtualisasi dengan *VMWare Fusion 7* dan *Mac OS X Yosemite 10.10.3*.
   b. Perangkat pengembang: Python 2.7 dan *MATLAB R2014b*.

4.2 Penjelasan Implementasi

Pada subbab ini akan dijelaskan parameter yang digunakan dan implementasi setiap subbab yang terdapat pada bab sebelumnya yaitu bab perancangan program. Pada bagian implementasi ini juga akan dijelaskan mengenai fungsi-fungsi yang digunakan dalam program tugas akhir ini dan disertai dengan kode sumber masing-masing fungsi utama.

4.3 Implementasi *Data Capture*

Implementasi Data Capture dilakukan sesuai desain program yang ditunjukkan pada Gambar 3.2 akan diaplikasikan menggunakan bahasa *python*. Yang dilakukan pertama kali adalah pembuatan socket untuk mengakses port dari NeuroSky MindWave agar program dapat menerima nilai sinyal. Bagian program ini sangat penting karena semua data yang akan diolah berasal dari bagian ini. Kode sumber implementasi *data capture* dikutip dari buku Tugas Akhir Ahmad Hayam Brillian yang berjudul "Pengenalan Sandi Morse dari Sinyal *Electroencephalogram* yang Direkam dengan *NeuroSky MindWave*" tahun 2015.

```
1.  def connect(self, add = 'localhost', port = 13854):
2.      self.c_sock     =     socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)
3.      self.c_sock.connect((add, port))
4.      auth_data  =  {"appName":  self.appName,  "appKey":
    self.appKey}
5.      self.c_sock.send(json.dumps(auth_data))
6.      conf = self.c_sock.recv(8192)
7.      conf = self.c_sock.recv(8192)
8.      print 'Connected...'
9.      self.__record()
```

Kode Sumber 4.1 Fungsi pembuatan socket dan inisiasi

Kode Sumber 4.2 menunjukkan bagian program yang berfungsi menampung rekaman yang diterima oleh elektroda dalam bentuk json yang diterapkan pada baris 1 sampai 24 nantinya json ini akan diekstrak untuk mendapatkan nilai-nilai sinyal EEG yang nantinya akan diolah.

```
1.  def __read_data(self, a, b):
2.      while self.__recording:
3.              buff = self.c_sock.recv(8192)
4.              if self.__islogging:
5.              self.temp += buff
6.              curr_time = time()
7.                      parts = buff.split('\r')
8.                      for part in parts:
9.                              if part.startswith('{'):
```

```
10.                                     try:
11.                                             parseObj    =
    json.loads(part)
12.                                             for   key   in
    parseObj.keys():
13.                                                     if
    key == 'eSense' or key == 'eegPower':
14.
        for key2 in parseObj[key].keys():
15.
                if len(self.data[key2]) > 4000000:
16.
                        self.data[key2].pop(0)
17.
                self.data[key2].append((curr_time,
    parseObj[key][key2]))
18.                                                     else:
19.
        if len(self.data[key]) > 4000000:
20.
                self.data[key].pop(0)
21.
        self.data[key].append((curr_time, parseObj[key]))
22.                                     except:
23.                                             print key
24.             sleep(0.05)
25.
26. def __record(self):
27.     if not self.__recording:
28.             conf_data   =   {"enableRawOutput":   True,
    "format": "Json"}
29.             self.c_sock.send(json.dumps(conf_data))
30.
31.             self.__recording = True
32.             # self.__tstart = time()
33.             start_new_thread(self.__read_data, (0, 0))
34.     else:
35.             print 'Recording...'
36.
37. def rec_data(self):
38.     if not self.__islogging:
39.                     self.__init_data()
40.                     self.__islogging = True
41.                     self.__tstart = time()
42.             else:
43.                     print 'Recording in progress...'
```

Kode Sumber 4.2 Fungsi penerimaan data

Setelah proses penerimaan data selesai, selanjutnya sinyal disimpan dengan menjalankan bagian program yang tertera pada Kode Sumber 4.3. Bagian program ini akan menyimpan raw value, alpha, beta, theta, delta, dan gamma dari sinyal EEG yang diterima elektroda.

```
1.  def simpan(a, k):
2.      b = a.get_recorded(withtime=False)
3.      np.array(b).tofile(k + '_raw_value', ' ')
4.      b   =   a.get_recorded(withtime=False,   param   =
    'highAlpha')
5.      np.array(b).tofile(k + '_alpha', ' ')
6.      b   =   a.get_recorded(withtime=False,   param   =
    'highBeta')
7.      np.array(b).tofile(k + '_beta', ' ')
8.      b   =   a.get_recorded(withtime=False,   param   =
    'highGamma')
9.      np.array(b).tofile(k + '_gamma', ' ')
10.     b   =   a.get_recorded(withtime=False,   param   =
    'delta')
11.     np.array(b).tofile(k + '_delta', ' ')
12.     b   =   a.get_recorded(withtime=False,   param   =
    'theta')
13.     np.array(b).tofile(k + '_theta', ' ')
```

Kode Sumber 4.3 Fungsi simpan

4.4 Implementasi *Preprocessing*

Sesuai dengan rancangan diagram alur yang tertera pada Gambar 3.1, setelah tahap pengumpulan data, tahapan berikutnya adalah melakukan preprocessing. Gambar 3.3 menunjukan serangkaian proses untuk melakukan penghilangan *noise* dengan mengaplikasikan fungsi *Noise Removal* dan ekstraksi fitur pada fungsi *Feature Extraction.*

4.4.1 Implementasi *Noise Removal*

Proses penghilangan noise adalah salah satu proses yang sangat penting dalam pengerjan Tugas Akhir ini karena data yang akan diekstrak fiturnya menjadi optimal dan tidak mengandung noise. Kode Sumber 4.4 akan menunjukkan bagaimana

implementasi fungsi ini sesuai dengan *pseudocode* yang tertera pada Gambar 3.4.

Pada tahapan rancangan disebutkan bahwa sinyal akan diaplikasikan fungsi FFT yang kalkulasinya terdapat pada baris3, sedangkan pada baris 7 akan dilakukan pemotongan. Untuk tujuan visualisasi, penerapan *invers transform* pada baris 8 mengembalikan sinyal ke domain waktu.

```
1.  function feat = FeatureExtraction(x)
2.  a = load(x);
3.  af = fft(a);
4.  N = length(a);
5.
6.  ax = af;
7.  af(25:length(af)) = 0;
8.  ai = real(ifft(af));
9.  plot(ai);
```

Kode Sumber 4.4 Fungsi Noise Removal

4.4.2 Implementasi *Feature Extraction*

Setelah data sinyal bersih dari *noise* selanjutnya adalah mengambil nilai *power spectrum* dari gelombang *alpha*, *beta*, *delta*, dan *theta* sesuai dengan rentang frekuensi gelombang dalam Tabel 2.2. *Power spectrum* ini didapat dengan menjumlah gelombang dalam masing-masing rentang frekuensi untuk selanjutnya dijadikan fitur dalam tahapan selanjutnya.

Langkah-langkah implementasi fungsi *feature extraction* ini ditampilkan dalam Kode Sumber 4.5. Baris 1 – 5 ditujukan untuk operasi penghilangan magnitude. Penjumlahan power spectrum untuk masing-masing gelombang akan ditunjukkan pada bari 7 sampai baris 14. Fitur *power spectrum* dari tiap gelombang akan disimpan dalam *array* dengan menjalankan baris 16.

```
spect = abs(ax).^2;
hspect = spect(1:(length(spect)/2));
N = length(a);
fs = 512;
hspect(25*N/fs:length(hspect)) = 0;

wd = hspect((0.5)*N/fs:4*N/fs);
f1 = sum(wd);
wt = hspect(4*N/fs:8*N/fs);
f2 = sum(wt);
wa = hspect(8*N/fs:13*N/fs);
f3 = sum(wa);
wb = hspect(13*N/fs:20*N/fs);
f4 = sum(wb);

feat = [delta,theta,alpha,beta]
```

Kode Sumber 4.5 Fungsi Feature Extraction

```
function total = extractor()

total = [];

files = getAllFiles('./selected');
files = files(2:end);

for i=1:length(files)
list = files(i);
c = list{1};
kelas = 1;
if c(end-2:end-2) == 'f'
kelas = 0;
else
kelas = 1;
end
feat = FeatureExtraction(list{1});
feat = [feat kelas];
total = [total; feat];
end
```

Kode Sumber 4.6 Fungsi Extractor

4.4.3 Implementasi Extractor

Untuk mendapatkan fitur dari seluruh gelombang EEG secara otomatis, perlu dilakukan pembacan direktori tempat menampung file sinyal EEG. Kode Sumber 4.6 menunjukkan implementasi fungsi *Extractor* dimana baris 3 menunjukkan inisiasi *vector* untuk menampung fitur sinyal yang telah diekstrak. Sedangkan dengan menjalankan baris ke 3 dan 4, hasil yang akan didapatkan adalah daftar *file* sinyal yang ada dalam direktori. Baris 8 – 16 akan memilih satu per satu file yang akan diekstrak dengan menjalankan baris 17 yang akan memanggil fungsi *Feature Extraction*. Sinyal yang telah diekstrak fiturnya akan disimpan ke dalam *vector* beserta kelasnya melalui fungsi pada baris 18-20.

4.5 Implementasi Klasifikasi

Pada tahapan implementasi klasifikasi, data sinyal EEG yang sudah diekstrak fiturnya akan dicocokkan kelasnya menggunakan fungsi SVMTrain dan SVMClassify. Tetapi, sebelum kedua fungsi tersebut bisa dijalankan, data fitur terlebih dulu dibagi menjadi dua. Pembagian data tersebut akan menjadi data training sebesar 75% dari keseluruhan data untuk menjalankan fungsi SVMTrain dan data testing sebanyak 25% data untuk menerapkan fungsi SVMClassify.

4.5.1 Implementasi SVMTrain

Eksekusi fungsi SVMTrain dilakukan secara bersamaan dengan menerapkan empat macam kernel dengan acuan beberap kernel yang terdapat pada Tabel 2.3 seperti *radial basis function (rbf)* ataupun *polynomial*. *Kernel linear* sendiri merupakan *kernel default* dari fungsi SVMTrain. *Kernel quadratic* adalah fungsi *kernel polynomial* dengan derajat 2. [**15**] Pada fungsi ini juga diaplikasikan *kernel polynomial* dengan derajat *default*, yaitu 3. Implementasi fungsi SVMTrain dijelaskan pada Kode Sumber 4.7. Pada kode sumber tersebut menunjukkan proses proses pengacakan data pada baris 1 dan 2 dan pembagian data pada

baris 4 – 11 dan proses pembuatan model yang terdapat pada baris 13 – 19 dengan sekaligus menjalankan empat jenis kernel SVM yang nantinya digunakan sebagai perbandingan akurasi.

```
1.  new = randperm(40);
2.  randdata = ans(new,:);
3.
4.  datatraining = randdata(1:30,:);
5.  datatesting = randdata(31:40,:);
6.
7.  datatrain = datatraining(:, 1:end-1);
8.  kelastrain = datatraining(:, end);
9.
10. datatest = datatesting(:, 1:end-1);
11. kelastest = datatesting(:, end);
12.
13. SVMStruct1 =
    svmtrain(datatrain,kelastrain,'kernel_function','l
    inear');
14.
15. SVMStruct2 =
    svmtrain(datatrain,kelastrain,'kernel_function','r
    bf');
16.
17. SVMStruct3 =
    svmtrain(datatrain,kelastrain,'kernel_function','q
    uadratic');
18.
19. SVMStruct4 =
    svmtrain(datatrain,kelastrain,'kernel_function','p
    olynomial');
```

Kode Sumber 4.7 Fungsi SVMTrain

4.5.2 Implementasi SVMClassify

Data hasil pembagian dengan data training sebesar 25% selanjutnya akan menjadi data testing untuk menguji *classifier/hyperplane* sebagai hasil training untuk mendapatkan akurasi dengan menjalankan fungsi SVMClassify yang ditunjukkan oleh Kode Sumber 4.8.

```
1.  classres1 = svmclassify(SVMStruct1,datatest);
2.
3.  diffclass1 = classres1 == kelastest;
4.  acc1 = sum(diffclass1);
5.
6.  confmat1 = zeros(2,2);
7.  for i=1:10
8.  a = classres1(i)+1;
9.  b = kelastest(i)+1;
10. confmat1(a,b) = confmat1(a,b)+1;
11. end
12.
13. classres2 = svmclassify(SVMStruct2,datatest);
14.
15. diffclass2 = classres2 == kelastest;
16. acc3 = sum(diffclass2);
17.
18. confmat2 = zeros(2,2);
19. for i=1:10
20. a = classres2(i)+1;
21. b = kelastest(i)+1;
22. confmat2(a,b) = confmat2(a,b)+1;
23. end
24.
25. classres3 = svmclassify(SVMStruct3,datatest);
26.
27. diffclass3 = classres3 == kelastest;
28. acc3 = sum(diffclass3);
29.
30. confmat2 = zeros(2,2);
31. for i=1:10
32. a = classres2(i)+1;
33. b = kelastest(i)+1;
34. confmat2(a,b) = confmat2(a,b)+1;
35. end
36.
37. classres4 = svmclassify(SVMStruct4,datatest);
38.
39. diffclass4 = classres4 == kelastest;
40. acc4 = sum(diffclass4);
41.
42. confmat4 = zeros(2,2);
43. for i=1:10
44. a = classres4(i)+1;
```

```
45. b = kelastest(i)+1;
46. confmat4(a,b) = confmat4(a,b)+1;
47. end
```

Kode Sumber 4.8 Fungsi SVMClassify

Model-model yang sudah dihasilkan fungsi sebelumnya akan dijalankan untuk masing-masing *data testing* dengan menjalankan fungsi ini yang ditunjukkan pada baris 1, 13, 25, dan 37. Sedangkan untuk penghitungan akurasi dan pembuatan *confussion matrix* ditunjukkan pada baris 6 – 11, 18 – 23, 30 – 35, dan 42 – 47.

```
1.  countNum = size(total,2);
2.
3.  for i = 1 : countNum
4.  data = total;
5.  end
6.  groups = ismember(data(:,5),1);
7.  k=8;
8.
9.  cvFolds = crossvalind('Kfold', groups, k);
10. cp = classperf(groups);
11.
12. for i = 1:k
13. testIdx = (cvFolds == i);
14. trainIdx = ~testIdx;
15.
16. svmStruct =
    svmtrain(data(trainIdx,1:4),groups(trainIdx),'k
    ernel_function','linear');
17.
18. res = svmclassify(svmStruct,
    data(testIdx,1:4));
19.
20. cp = classperf(cp, res, testIdx);
21. end
22.
23. confmat = cp.CountingMatrix;
24. accuracy = cp.CorrectRate;
```

Kode Sumber 4.9 Fungsi kCVal

4.5.3 Implementasi kCVal

Seluruh data yang sudah didapat dari fungsi *feature extraction* akan dipartisi sebanyak *k* lalu data ke *k* akan digunakan sebagai data *testing* data ke 1 sampai *k – 1* akan digunakan sebagai data *training*. Implementasi fungsi ini ditunjukkan pada Kode Sumber 4.9. Pada baris 1 – 6 dilakukan pembacaan data, mulai dari ukuran dataset hingga pemilihan kolom kelas dari dataset. Baris 7 menunjukkan nilai *k* yang akan digunakan. Pembuatan index untuk partisi data ditunjukkan pada baris 9 dan baris 10 untuk mengitung performa validasi. Baris 12 – 21 akan melakukan pemilihan subset yang akan dilatih dan diuji menggunakan SVM dengan *linear kernel*. *Confusion matrix* dan akurasi akan didapatkan setelah baris 23 dan 24 dijalankan.

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan uji coba yang dilakukan pada aplikasi yang telah dikerjakan serta analisa dari uji coba yang telah dilakukan. Pembahasan pengujian meliputi lingkungan uji coba, skenario uji coba yang meliputi uji kebenaran dan uji kinerja serta analisa setiap pengujian.

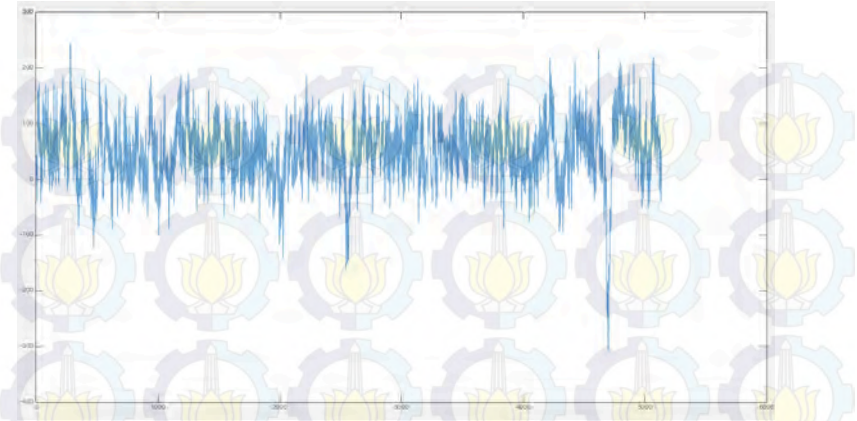
5.1 Lingkungan Uji Coba

Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menguji implementasi metode Transformasi *Fourier* dan *Support Vector* Machine pada Tugas Akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

3. Perangkat keras
   a. Prosesor: Intel® Core™ i5-5250U CPU @ 1.60GHz
   b. *Memory*(RAM): 4,00 GB
   c. Tipe Sistem: 64-bit Sistem operasi
4. Perangkat lunak
   a. Sistem operasi: *Windows 8.1 Pro* tervirtualisasi dengan *VMWare Fusion 7* dan *Mac OS X Yosemite 10.10.3*.
   b. Perangkat pengembang: Python 2.7 dan MATLAB R2014b.

5.2 Data Uji

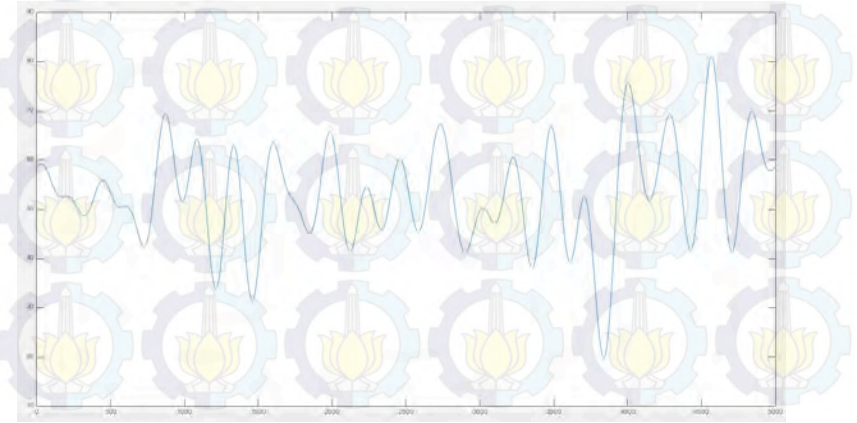
Data yang akan diuji merupakan data hasil rekaman menggunakan NeuroSky MindWave sesuai kondisi yang sudah dijelaskan pada subbab 2.2. Gambar 5.1 akan menunjukkan contoh visualisasi hasil rekaman EEG. Rekaman ini memiliki frekuensi *sampling* sebesar 512 Hz, dengan durasi 10 detik maka satu file terdiri atas ± 5000 nilai.



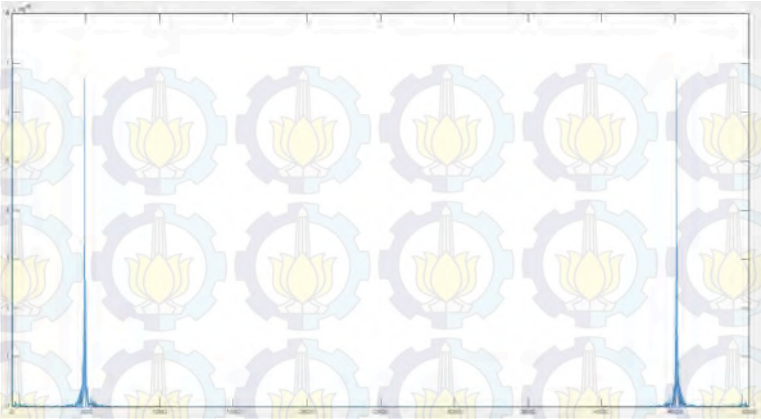
Gambar 5.1 Contoh rekaman EEG

5.3 Pemrosesan Data

Semua data hasil rekaman akan di proses pada tahapan preprocessing menjalankan Kode Sumber 4.4 dan Kode Sumber 4.5 untuk menghilangkan noise dan mendapatkan fiturnya dari sinyal EEG. Hasil dari running fungsi noise removal tersebut akan ditampilkan pada Gambar 5.2.



Gambar 5.2 Contoh hasil *noise removal*



Gambar 5.3 Contoh hasil penghapusan *magnitude*

Sebelum gelombang EEG dapat dipilih frekuensi sinyal dan diambil nilai *power spectrum*-nya, perlu dihilangkan *magnitude*-nya. Gambar 5.3 menunjukkan gelombang hasil transformasi *fourier* yang sudah siap untuk proses pemilihan sinyal dan kalkulasi *power spectrum*. Hasil operasi tersebut ditampilkan pada Tabel 5.1.

Tabel 5.1 Contoh nilai fitur sinyal EEG

Jenis Gelombang	Nilai
delta	10489375298.4080
theta	11411547001.8448
alpha	5676671238.27336
beta	5966074854.15926

5.4 Data *Testing* dan *Training*

Dalam pengujan ini, tahapan klasifikasi menggunakan *dataset* fitur sinyal EEG dengan *percentage split* sebesar 75% untuk *data training* dan 25% untuk *data testing*. Selain dengan

percentage split, penghitungan akurasi juga diterapkan dengan *cross validation* dengan nilai *fold* sebagai skenario uji coba. Keseluruhan *dataset* dengan 40 *instance* ditunjukkan pada Tabel 5.2. Kelas dengan nilai 1 menunjukkan data normal, sedangkan 0 menunjukkan data *fatigue*.

Tabel 5.2 Dataset sinyal EEG

delta	theta	alpha	beta	class
12203618983	6955171046	4252237268	2362493701	1
6861136625	6573293221	6041853826	6411416285	1
8482967199	6634503430	3276275656	1520138905	1
51394054600	37921953897	23798352314	9819334928	1
10330658097	7916013318	4537835851	2833928911	1
4716821477	4030418880	3328768161	2467382549	1
14893056650	7192625352	3389639825	2201298835	1
12154156505	11320710502	12121386615	7388475499	1
7606083521	20780075652	12590798580	14237051796	1
23065546125	32873310780	19045062943	10320215001	1
12883111235	14562303702	9364188370	2826096735	1
8126357423	6900505114	4381765012	1865086489	1
16346878992	56309495208	18768564230	10131845365	1
19812825319	16361611882	23821934055	7605661339	1
28913474048	27902838613	25656712630	22321677534	1
11895101757	9269442559	5245744451	3068392594	1
19261262302	14206914035	6189768711	3410134097	1
11626640795	12579325867	5108812293	2773001927	1
12209036419	12565895230	6341221440	3437535853	1
27826573702	21075947322	12403783831	9348841731	1
15209255432	15145559440	18524937244	26229940729	0
7024420089	5164797416	11904060204	7210779266	0

delta	theta	alpha	beta	class
6154953823	5784531965	9874686709	7183688191	0
9152131287	9213156593	11425914585	8133859107	0
10883157244	4623412835	11552248548	2680756082	0
6514064542	6441781318	10391330325	1835223965	0
1282730086	983894688,4	3070817731	1235848830	0
1467265534	1675660439	12792485753	1699733633	0
1807273433	1853424072	12502188376	1342525286	0
6099069077	3150997845	2511038257	1411477298	0
4958199135	2807500565	11474602504	1523120797	0
1953101488	2829624424	9662423263	1341771436	0
1738711517	2903681429	8219873470	1154818420	0
1338128128	2261884375	9318037280	1258125478	0
1797525814	2203379922	12730221677	1101530333	0
1818196067	1848137952	10204225144	1147982272	0
3612994245	2478579508	14366549029	1371605348	0
2151836377	2473095778	9226762484	1680565640	0
2391016074	1675850800	13465338962	2001935963	0
10511057728	11398667599	5680675189	5952043492	0

5.5 Skenario dan Evaluasi Pengujian

Uji coba ini dilakukan untuk menguji apakah fungsionalitas program telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya. Uji coba akan didasarkan pada beberapa skenario untuk menguji kesesuaian dan kinerja aplikasi.

Skenario pengujian terdiri dari 2 pengujian yaitu:

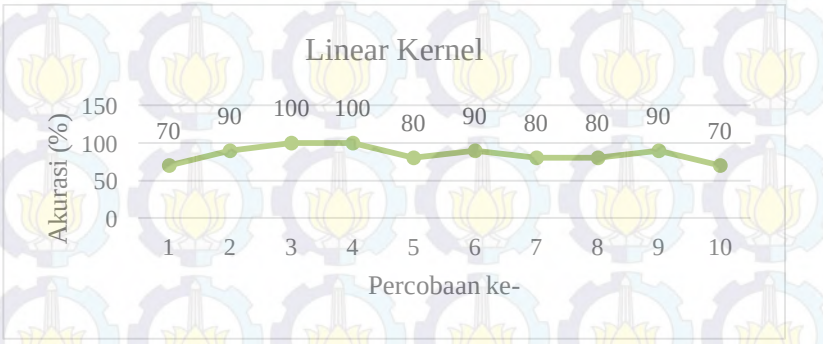
1. Skenario perhitungan akurasi dengan perbandingan 4 macam kernel SVM.
2. Skenario perhitungan akurasi menggunakan *linear kernel* SVM dengan parameter *k* pada *k-fold cross validation*.

5.5.1 Skenario Uji Coba 1

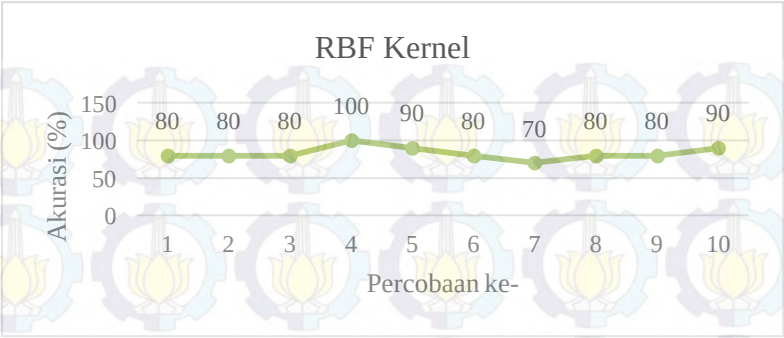
Skenario uji coba 1 adalah penghitungan akurasi dengan 4 macam *kernel* SVM, yaitu *linear*, *rbf*, *quadratic*, dan *polynomial kernel*. Uji coba ini dilakukan untuk mencari kernel mana yang paling sesuai digunakan untuk *dataset* pada Tabel 5.2 dan metode klasifikasi SVM. Uji coba ini akan dilakukan dengan melakukan *running* sebanyak 10 kali dan pengacakan ulang keseluruhan data sebelum dilakukan *splitting* untuk *training* dan *testing* setiap kali *running*.

Setelah dilakukan percobaan, maka harus dilakukan evaluasi untuk melihat sejauh mana kesesuaian program dengan data *testing* yang disediakan. Untuk menentukan kesesuaian keluaran program dan analisis yang telah dibuat dibutuhkan penghitungan akurasi. Penghitungan akurasi dilakukan dengan mencocokkan hasil keluaran program dengan *ground truth* yang dimiliki setiap *instance*.

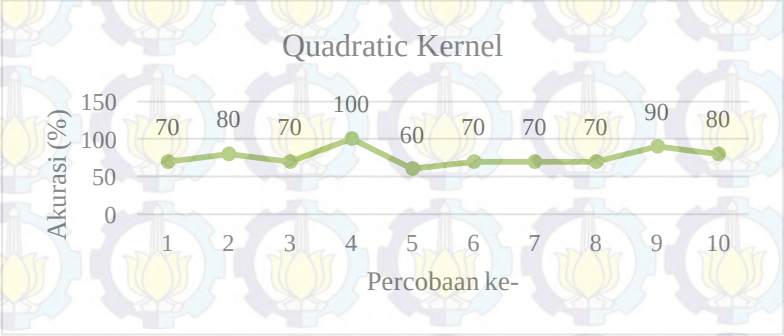
Untuk mempermudah analisis, hasil akurasi setiap kernel akan ditunjukkan dalam bentuk grafik yang bisa dilihat pada Gambar 5.4, Gambar 5.5, Gambar 5.6, dan Gambar 5.7. Rata-rata hasil akurasi setiap kernel juga disajikan pada Tabel 5.3.



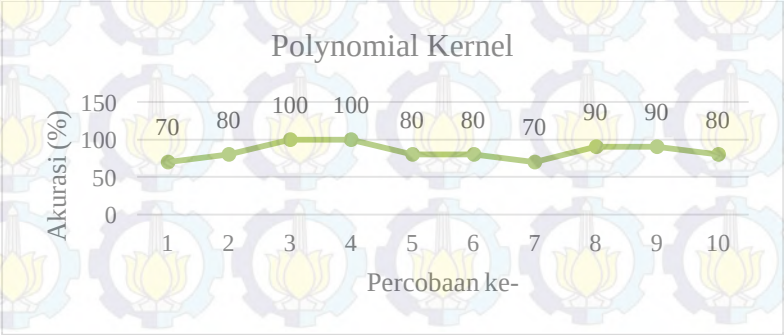
Gambar 5.4 Grafik akurasi *linear kernel* pada Skenario 1



Gambar 5.5 Grafik akurasi *RBF kernel* Skenario 1



Gambar 5.6 Grafik akurasi *quadratic kernel* Skenario 1



Gambar 5.7 Grafik akurasi *polynomial kernel* Skenario 1

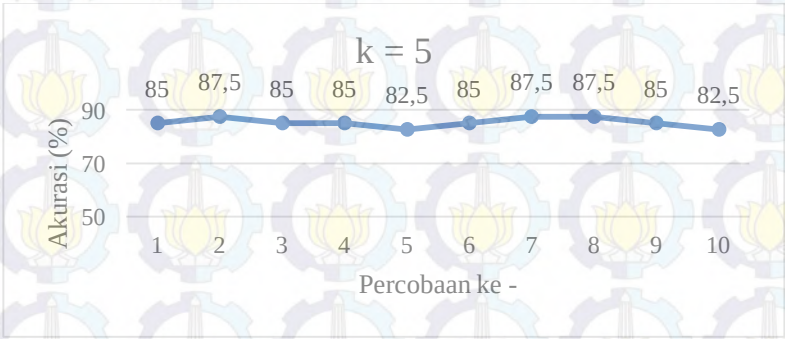
Tabel 5.3 Rata-rata akurasi kernel

Kernel	Rata-rata akurasi(%)
linear	**85**
rbf	83
quadratic	76
polynomial	84

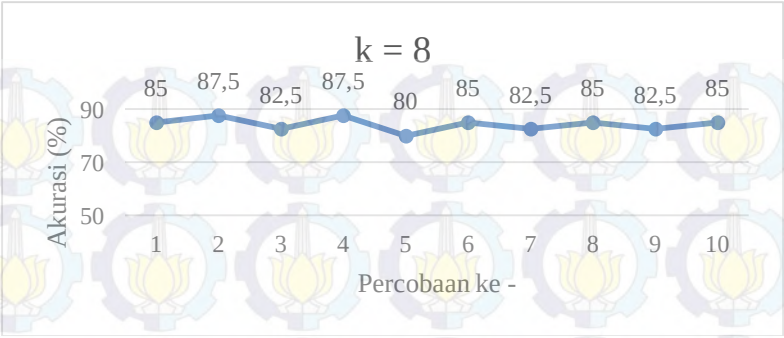
Pada akhir percobaan, dapat disimpulkan bahwa kinerja linear kernel menunjukkan akurasi yang paling tinggi dibanding tiga kernel lainnya, yaitu sebesar 85%. Hasil skenario uji coba 1 secara lengkap dapat dilihat pada LAMPIRAN.

5.5.2 Skenario Uji Coba 2

Skenario uji coba 2 merupakan uji coba penghitungan akurasi dengan menggunakan *trial and error basis* untuk parameter *k* pada *k-fold cross validation* dan SVM dengan *linear kernel*. Proses validasi ini dilakukan untuk menguji setiap data yang ada terhadap sistem dengan mempartisi data menjadi subset sejumlah *k*. Masing-masing subset hanya digunakan untuk sekali testing.

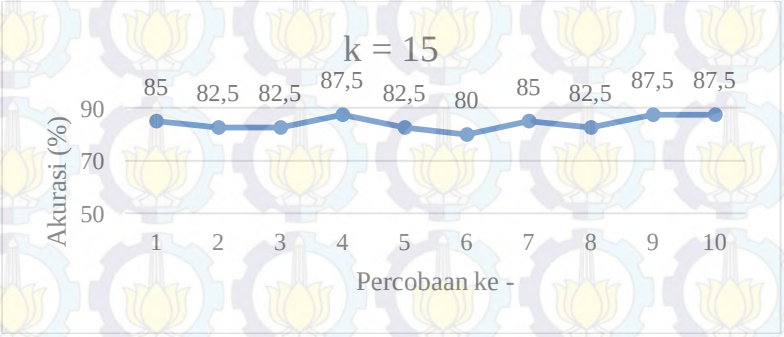


Gambar 5.8 Grafik akurasi 5-fold Skenario 2

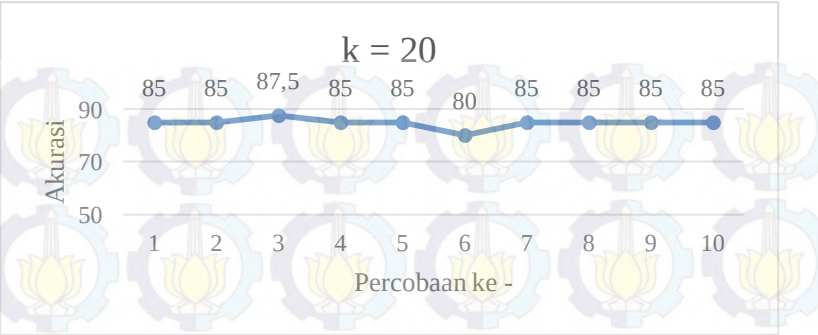


Gambar 5.9 Grafik akurasi 8-fold Skenario 2

Gambar 5.10 Grafik akurasi 10-fold Skenario 2



Gambar 5.11 Grafik akurasi 15-fold Skenario 2



Gambar 5.12 Grafik akurasi 20-fold Skenario 2

Tabel 5.4 Akurasi berdasarkan nilai *k* pada *k-fold Cross Validation*

Nilai *k*	Akurasi(%)
5	**85**
8	84
10	85
15	84
20	85

Tabel 5.4 menunjukkan bahwa percobaan menghasilkan nilai *k* terbaik untuk metode *k-fold cross validation* adalah 5, 10, dan 20 dengan akurasi 85%. Hasil skenario uji coba 2 secara lengkap dapat dilihat pada LAMPIRAN.

5.6 Analisis Hasil Uji Coba

Dari hasil skenario uji coba 1 didapatkan hasil klasifikasi terbaik dengan rata-rata akurasi sebesar 85% dengan menggunakan *linear kernel*. Hal ini menjadi salah satu faktor untuk penggunaan kernel yang sama pada skenario uji coba 2. Penerapan metode *k-fold* untuk melakukan validasi dataset

menunjukkan akurasi yang sebesar 85% dengan nilai k = 5. Nilai k yang diujicobakan tidak begitu menunjukkan perbedaan yang signifikan antara satu sama lain dengan selisih 1%.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap sinyal EEG untuk menentukan lelah atau tidaknya kondisi mental seseorang dapat diambil kesimpulan sebagai berikut:

1. Metode transformasi *fourier* atau yang digunakan disini adalah FFT sudah mampu mengurai sinyal untuk didapatkan fitur-fiturnya agar dapat diklasifikasi dengan baik.
2. Metode *Support Vector Machine* yang digunakan kombinasi kernel mampu melakukan klasifikasi terhadap *dataset* yang disediakan.
3. Berdasarkan hasil uji coba, dihasilkan akurasi terbaik pembagian data 3 : 1 dengan nilai rata-rata 85% dengan menggunakan *linear kernel*.
4. Sedangkan untuk hasil uji coba dengan menggunakan *k-fold cross validation* didapatkan nilai $k = 5$ dengan akurasi 85%.

6.2 Saran

Saran yang diberikan untuk pengembangan aplikasi ini adalah:

1. Alat yang digunakan untuk rekaman sinyal EEG ada baiknya ditingkatkan jumlah elektrodanya, karena alat yang digunakan pada Tugas Akhir ini hanya memiliki elektroda dibagian depan dan sulit untuk melakukan rekaman pada bagian kulit kepala lainnya.

2. Ada baiknya dibuatkan *user interface* pada tahapan klasifikasi untuk mempermudah pengguna dalam memasukkan nilai maupun melakukan pengamatan hasil pada tahapan tersebut.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap sinyal EEG untuk menentukan lelah atau tidaknya kondisi mental seseorang dapat diambil kesimpulan sebagai berikut:

1. Metode transformasi *fourier* atau yang digunakan disini adalah FFT sudah mampu mengurai sinyal untuk didapatkan fitur-fiturnya agar dapat diklasifikasi dengan baik.
2. Metode *Support Vector Machine* yang digunakan kombinasi kernel mampu melakukan klasifikasi terhadap *dataset* yang disediakan.
3. Berdasarkan hasil uji coba, dihasilkan akurasi terbaik pembagian data 3 : 1 dengan nilai rata-rata 85% dengan menggunakan *linear kernel*.
4. Sedangkan untuk hasil uji coba dengan menggunakan *k-fold cross validation* didapatkan nilai $k = 5$ dengan akurasi 85%.

6.2 Saran

Saran yang diberikan untuk pengembangan aplikasi ini adalah:

1. Alat yang digunakan untuk rekaman sinyal EEG ada baiknya ditingkatkan jumlah elektrodanya, karena alat yang digunakan pada Tugas Akhir ini hanya memiliki elektroda dibagian depan dan sulit untuk melakukan rekaman pada bagian kulit kepala lainnya.

2. Ada baiknya dibuatkan *user interface* pada tahapan klasifikasi untuk mempermudah pengguna dalam memasukkan nilai maupun melakukan pengamatan hasil pada tahapan tersebut.

LAMPIRAN

Tabel A.1 Rekap Hasil Keluaran Program Skenario Uji Coba 1 Percobaan 1

Data Testing					Kernel			
Delta	Theta	Alpha	Beta	Class	Linear	Rbf	Quadratic	Polynomial
12203618983	6955171046	4252237268	2362493701	1	1	1	1	1
6154953823	5784531965	9874686709	7183688191	0	0	1	1	1
19812825319	16361611882	23821934055	7605661339	1	0	1	1	1
10883157244	4623412835	11552248548	2680756082	0	0	0	0	0
27826573702	21075947322	12403783831	9348841731	1	1	1	1	1
10511057728	11398667599	5680675189	5952043492	0	1	1	1	1
1738711517	2903681429	8219873470	1154818420	0	0	0	0	0
7606083521	20780075652	12590798580	14237051796	1	0	1	1	0
7024420089	5164797416	11904060204	7210779266	0	0	0	1	0
2391016074	1675850800	13465338962	2001935963	0	0	0	0	0
				Accuracy	70%	80%	70%	70%

Tabel A.2 Rekap Hasil Keluaran Program Skenario Uji Coba 1 Percobaan 2

Data Testing					Kernel			
Delta	Theta	Alpha	Beta	Class	Linear	Rbf	Quadratic	Polynomial
12883111235	14562303702	9364188370	2826096735	1	1	1	1	1
2391016074	1675850800	13465338962	2001935963	0	0	0	0	0
12154156505	11320710502	12121386615	7388475499	1	1	0	0	0
14893056650	7192625352	3389639825	2201298835	1	1	1	1	1
7024420089	5164797416	11904060204	7210779266	0	0	0	0	0
1467265534	1675660439	12792485753	1699733633	0	0	0	0	0
1953101488	2829624424	9662423263	1341771436	0	0	0	0	0
1338128128	2261884375	9318037280	1258125478	0	0	0	0	0
2151836377	2473095778	9226762484	1680565640	0	0	0	0	0
6861136625	6573293221	6041853826	6411416285	1	0	0	0	0
				Accuracy	90%	80%	80%	80%

Tabel A.3 Rekap Hasil Keluaran Program Skenario Uji Coba 1 Percobaan 3

Data Testing					Kernel			
Delta	Theta	Alpha	Beta	Class	Linear	Rbf	Quadratic	Polynomial
27826573702	21075947322	12403783831	9348841731	1	1	1	1	1
10330658097	7916013318	4537835851	2833928911	1	1	1	1	1
2391016074	1675850800	13465338962	2001935963	0	0	0	0	0
14893056650	7192625352	3389639825	2201298835	1	1	1	1	1
6154953823	5784531965	9874686709	7183688191	0	0	1	1	0
12203618983	6955171046	4252237268	2362493701	1	1	1	1	1
9152131287	9213156593	11425914585	8133859107	0	0	1	1	0
28913474048	27902838613	25656712630	22321677534	1	1	1	0	1
1953101488	2829624424	9662423263	1341771436	0	0	0	0	0
19812825319	16361611882	23821934055	7605661339	1	0	1	1	1
				Accuracy	100%	80%	70%	100%

Tabel A.4 Rekap Hasil Keluaran Program Skenario Uji Coba 1 Percobaan 4

Data Testing					Kernel			
Delta	Theta	Alpha	Beta	Class	Linear	Rbf	Quadratic	Polynomial
4958199135	2807500565	11474602504	1523120797	0	0	0	0	0
9152131287	9213156593	11425914585	8133859107	0	0	0	0	0
1467265534	1675660439	12792485753	1699733633	0	0	0	0	0
1738711517	2903681429	8219873470	1154818420	0	0	0	0	0
1953101488	2829624424	9662423263	1341771436	0	0	0	0	0
12883111235	14562303702	9364188370	2826096735	1	1	1	1	1
23065546125	32873310780	19045062943	10320215001	1	1	1	1	1
1797525814	2203379922	12730221677	1101530333	0	0	0	0	0
8482967199	6634503430	3276275656	1520138905	1	1	1	1	1
6514064542	6441781318	10391330325	1835223965	0	0	0	0	0
				Accuracy	100%	100%	100%	100%

Tabel A.5 Rekap Hasil Keluaran Program Skenario Uji Coba 1 Percobaan 5

Data Testing					Kernel			
Delta	Theta	Alpha	Beta	Class	Linear	Rbf	Quadratic	Polynomial
10330658097	7916013318	4537835851	2833928911	1	1	1	1	1
1467265534	1675660439	12792485753	1699733633	0	0	0	0	0
8126357423	6900505114	4381765012	1865086489	1	1	1	1	1
4716821477	4030418880	3328768161	2467382549	1	0	0	0	0
9152131287	9213156593	11425914585	8133859107	0	0	0	1	0
6514064542	6441781318	10391330325	1835223965	0	0	0	0	0
15209255432	15145559440	18524937244	26229940729	0	0	1	1	1
16346878992	56309495208	18768564230	10131845365	1	1	1	0	1
2151836377	2473095778	9226762484	1680565640	0	0	0	0	0
28913474048	27902838613	25656712630	22321677534	1	1	1	1	1
				Accuracy	90%	80%	70%	80%

Tabel A.6 Rekap Hasil Keluaran Program Skenario Uji Coba 1 Percobaan 6

Data Testing					Kernel			
Delta	Theta	Alpha	Beta	Class	Linear	Rbf	Quadratic	Polynomial
4716821477	4030418880	3328768161	2467382549	1	1	1	0	0
19261262302	14206914035	6189768711	3410134097	1	1	1	1	1
8482967199	6634503430	3276275656	1520138905	1	1	1	1	1
16346878992	56309495208	18768564230	10131845365	1	1	1	1	1
23065546125	32873310780	19045062943	10320215001	1	1	1	1	1
12154156505	11320710502	12121386615	7388475499	1	0	0	0	0
19812825319	16361611882	23821934055	7605661339	1	0	1	0	1
28913474048	27902838613	25656712630	22321677534	1	1	1	0	1
4958199135	2807500565	11474602504	1523120797	0	0	0	0	0
1807273433	1853424072	12502188376	1342525286	0	0	0	0	0
				Accuracy	80%	90%	60%	80%

Tabel A.7 Rekap Hasil Keluaran Program Skenario Uji Coba 1 Percobaan 7

Data Testing					Kernel			
Delta	Theta	Alpha	Beta	Class	Linear	Rbf	Quadratic	Polynomial
11895101757	9269442559	5245744451	3068392594	1	1	1	1	1
14893056650	7192625352	3389639825	2201298835	1	1	1	1	1
6154953823	5784531965	9874686709	7183688191	0	0	0	0	0
28913474048	27902838613	25656712630	22321677534	1	1	1	0	1
8482967199	6634503430	3276275656	1520138905	1	1	1	1	1
12154156505	11320710502	12121386615	7388475499	1	1	0	0	0
8126357423	6900505114	4381765012	1865086489	1	1	1	1	1
4716821477	4030418880	3328768161	2467382549	1	0	0	1	0
10511057728	11398667599	5680675189	5952043492	0	1	1	1	1
1807273433	1853424072	12502188376	1342525286	0	0	0	0	0
				Accuracy	80%	70%	70%	70%

Tabel A.8 Rekap Hasil Keluaran Program Skenario Uji Coba 1 Percobaan 8

Data Testing					Kernel			
Delta	Theta	Alpha	Beta	Class	Linear	Rbf	Quadratic	Polynomial
11895101757	9269442559	5245744451	3068392594	1	1	1	1	1
6099069077	3150997845	2511038257	1411477298	0	1	1	1	1
12203618983	6955171046	4252237268	2362493701	1	1	1	1	1
51394054600	37921953897	23798352314	9819334928	1	1	1	0	1
1738711517	2903681429	8219873470	1154818420	0	0	0	0	0
19261262302	14206914035	6189768711	3410134097	1	1	1	1	1
11626640795	12579325867	5108812293	2773001927	1	1	1	1	1
9152131287	9213156593	11425914585	8133859107	0	0	1	0	0
10883157244	4623412835	11552248548	2680756082	0	1	0	0	0
4958199135	2807500565	11474602504	1523120797	0	0	0	0	0
				Accuracy	80%	80%	70%	90%

Tabel A.9 Rekap Hasil Keluaran Program Skenario Uji Coba 1 Percobaan 9

Data Testing					Kernel			
Delta	Theta	Alpha	Beta	Class	Linear	Rbf	Quadratic	Polynomial
3612994245	2478579508	14366549029	1371605348	0	0	0	0	0
1818196067	1848137952	10204225144	1147982272	0	0	0	0	0
6514064542	6441781318	10391330325	1835223965	0	0	0	0	0
10511057728	11398667599	5680675189	5952043492	0	1	1	1	1
9152131287	9213156593	11425914585	8133859107	0	0	1	0	0
11895101757	9269442559	5245744451	3068392594	1	1	1	1	1
1338128128	2261884375	9318037280	1258125478	0	0	0	0	0
27826573702	21075947322	12403783831	9348841731	1	1	1	1	1
8126357423	6900505114	4381765012	1865086489	1	1	1	1	1
14893056650	7192625352	3389639825	2201298835	1	1	1	1	1
				Accuracy	90%	80%	90%	90%

Tabel A.10 Rekap Hasil Keluaran Program Skenario Uji Coba 1 Percobaan 10

Data Testing					Kernel			
Delta	Theta	Alpha	Beta	Class	Linear	Rbf	Quadratic	Polynomial
10330658097	7916013318	4537835851	2833928911	1	1	1	1	1
2391016074	1675850800	13465338962	2001935963	0	0	0	0	0
1338128128	2261884375	9318037280	1258125478	0	0	0	0	0
2151836377	2473095778	9226762484	1680565640	0	0	0	0	0
7606083521	20780075652	12590798580	14237051796	1	0	1	0	0
19812825319	16361611882	23821934055	7605661339	1	0	1	1	1
1467265534	1675660439	12792485753	1699733633	0	0	0	0	0
6099069077	3150997845	2511038257	1411477298	0	1	1	1	1
11626640795	12579325867	5108812293	2773001927	1	1	1	1	1
10883157244	4623412835	11552248548	2680756082	0	0	0	0	0
				Accuracy	70%	90%	80%	80%

Tabel A.11 Rekap Hasil Skenario Uji Coba 2 Percobaan 1

k = 5			
Running	Confusion Matrix		Accuracy
r1	17	3	85%
	3	17	
r2	17	2	88%
	3	18	
r3	17	3	85%
	3	17	
r4	18	4	85%
	2	16	
r5	16	3	83%
	4	17	
r6	17	3	85%
	3	17	
r7	18	3	88%
	2	17	
r8	18	3	88%
	2	17	
r9	18	4	85%
	2	16	
r10	17	4	83%
	3	16	
	Average Accuracy		85%

Tabel A.12 Rekap Hasil Skenario Uji Coba 2 Percobaan 2

$k = 8$			
Running	Confusion Matrix		Accuracy
r1	17	3	85%
	3	17	
r2	18	3	88%
	2	17	
r3	17	4	83%
	3	16	
r4	17	2	88%
	3	18	
r5	17	5	80%
	3	15	
r6	18	4	85%
	2	16	
r7	17	4	83%
	3	16	
r8	18	4	85%
	2	16	
r9	17	4	83%
	3	16	
r10	17	3	85%
	3	17	
	Average Accuracy		84%

Tabel A.13 Rekap Hasil Skenario Uji Coba 2 Percobaan 3

k = 10			
Running	Confusion Matrix		Accuracy
r1	17	2	88%
	3	18	
r2	18	4	85%
	2	16	
r3	17	4	83%
	3	16	
r4	17	3	85%
	3	17	
r5	17	4	83%
	3	16	
r6	18	3	88%
	2	17	
r7	18	4	85%
	2	16	
r8	19	4	88%
	1	16	
r9	17	4	83%
	3	16	
r10	17	3	85%
	3	17	
	Average Accuracy		85%

Tabel A.14 Rekap Hasil Skenario Uji Coba 2 Percobaan 4

k = 15			
Running	Confusion Matrix		Accuracy
r1	17	3	85%
	3	17	
r2	16	3	83%
	4	17	
r3	17	4	83%
	3	16	
r4	18	3	88%
	2	17	
r5	17	4	83%
	3	16	
r6	16	4	80%
	4	16	
r7	18	4	85%
	2	16	
r8	17	4	83%
	3	16	
r9	18	3	88%
	2	17	
r10	18	3	88%
	2	17	
	Average Accuracy		84%

Tabel A.15 Rekap Hasil Skenario Uji Coba 2 Percobaan 5

k = 20			
Running	Confusion Matrix		Accuracy
r1	17	3	85%
	3	17	
r2	17	3	85%
	3	17	
r3	18	3	88%
	2	17	
r4	17	3	85%
	3	17	
r5	17	3	85%
	3	17	
r6	16	4	80%
	4	16	
r7	17	3	85%
	3	17	
r8	17	3	85%
	3	17	
r9	17	3	85%
	3	17	
r10	17	3	85%
	3	17	
	Average Accuracy		85%

DAFTAR PUSTAKA

[6] westminster.edu. [Online]. http://www.psych.westminster.edu/psybio/BN/Labs/Brainwaves.htm

[3] Immuno Laboratories, Inc. (2010, May) Better Health USA. [Online]. http://www.betterhealthusa.com/public/235.cfm

[8] H. Kececi and Y. Degirmenci, "Quantitative EEG and Cognitive Evoked Potentials in Anemia," *Clinical Neurophysiology*, pp. 137-143, April 2008.

[11] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, p. 273, 1995.

[9] James W. Cooley and John W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297-301, 1965.

[10] J. W., P. Lewis and P. Welch Cooley, "The Fast Fourier Transform and its Applications," *IEEE Trans on Education*, vol. 12, no. 1, pp. 28-34, 1969.

[7] Kai-Quan Shen, Xiao-Ping Li, Chong-Jin Ong, Shi-Yun Shao, and Einar P. V. Wilder-Smith, "EEG-based mental fatigue measurement using multi-class support vector machines with confidence estimate," *International Federation of Clinical Neurophysiology*, pp. 1524-1533, 2008.

[4] N. Andharu F. P. et al., "Analyzing Brainwave Using Single Electroencephalographic Channel To Identify Manufacturing Supervisor Fatigue," in *International Conference on Information, Communication Technology and System*, Surabaya, 2014, pp. 87-92.

[2] Samuele M. Marcora, Walter Staiano, and Victoria Manning, "Mental fatigue impairs physical performance in humans," *Journal of Applied Physiology*, vol. 106, no. 3, pp. 857-864, March 2009.

[13] Anto Satriyo Nugroho, Arief Budi Witarto, and Dwi Handoko, "Support Vector Machine - Teori dan Aplikasinya dalam Bioinformatika," *Kuliah Umum IlmuKomputer.com*, 2003.

[15] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, *Introduction to Information Retrieval*, Online ed. Cambridge, England, United Kingdom: Cambridge University Press, 2009.

[5] ThinkGeek. [Online]. http://www.thinkgeek.com/product/e9e5/

[12] Statsoft. [Online]. http://www.statsoft.com/Textbook/Support-Vector-Machines

[1] DrowsyDriving.org. [Online]. http://drowsydriving.org/about/facts-and-stats/

[14] James McCaffrey. (2013, October) Visual Studio Magazine. [Online]. http://visualstudiomagazine.com/articles/2013/10/1/understanding-and-using-kfold.aspx

BIODATA PENULIS

Farras Kinan, lahir di Surabaya, pada tanggal 17 September 1993. Penulis menempuh pendidikan mulai dari SD Negeri Manukan Kulon V Surabaya (1999-2005), SMP Negeri 26 Surabaya (2005-2008), SMA Negeri 11 Surabaya Surabaya (2008-2011) dan S1 Teknik Informatika ITS (2011-2015).

Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC). Diantaranya adalah menjadi staff Departemen Pengembangan Profesi Himpunan Mahasiswa Teknik Computer ITS 2012-2013. Penulis juga aktif dalam kegiatan kepanitiaan Schematics. Diantaranya penulis pernah menjadi staff Desain, Dekorasi, dan Dokumentasi (3D) Schematics 2012 dan ketua divisi 3D Schematics 2013. Penulis pernah menjadi peserta sekaligus pemenang kompetisi GeMasTIK tahun 2012 dalam kategori Piranti Cerdas dan *Embedded System* pada tahun 2013.

Selama kuliah di teknik informatika ITS, penulis mengambil bidang minat Komputasi Cerdas dan Visi (KCV). Komunikasi dengan penulis dapat melalui email: **farras.kinan@live.com**.